Ph.D. Thesis

Integer Programming Models for Ground-Holding in Air Traffic Flow Management

by R. Hoffman
Advisor: M. Ball

NEXTOR Ph.D. 98-1
(ISR Ph.D. 98-4)

ABSTRACT

Title of Dissertation: INTEGER PROGRAMMING MODELS FOR GROUND-HOLDING IN AIR TRAFFIC FLOW MANAGEMENT

Robert L. Hoffman, Doctor of Philosophy, 1997

Dissertation directed by: Professor Michael O. Ball
Department of Management Sciences,
College of Business and Management

In this dissertation, integer programming models are applied to combinatorial problems in air traffic flow management. For the two problems studied, models are developed and analyzed both theoretically and computationally. This dissertation makes contributions to integer programming while providing efficient tools for solving air traffic flow management problems.

Currently, a constrained arrival capacity situation at an airport in the United States is alleviated by holding inbound aircraft at their departure gates. The ground holding problem (GH) decides which aircraft to hold on the ground and for how long. This dissertation examines the GH from two perspectives. First, the hubbing operations of the airlines are considered by adding side constraints to GH. These constraints enforce the desire of the airlines to temporally group banks of flights. Five basic models and several variations of the ground holding problem with banking constraints (GHB) are presented. A particularly strong, facet-inducing model of the banking constraints is presented which allows one to solve large instances of GHB in less than half-an-hour of CPU time.

Secondly, the stochastic nature of arrival capacity is modeled by an integer program that provides the optimal trade-off between ground delay and airborne delay. The dual network properties of the integer program allow one to obtain integer solutions directly from the linear programming relaxation.

This model is designed to work in close conjunction with the most recent operational paradigms developed by the joint venture between the FAA and the airlines known as collaborative decision making (CDM). Both these paradigms and the impact of CDM on the decision making process in air traffic flow management are thoroughly discussed.

The work on banking constraints analyzes several alternative formulations. It involves the use of auxiliary decision variables, the application of special branching techniques and the use of facet-inducing constraints. The net result is to reduce by several orders of magnitude the computation time and resources necessary to solve the integer program to optimality. The work on the stochastic ground holding problem shows that the model's underlying matrix is totally unimodular by transforming the dual into a network flow model.

INTEGER PROGRAMMING MODELS FOR GROUND-HOLDING IN AIR TRAFFIC FLOW MANAGEMENT

by

Robert L. Hoffman

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland at College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
1997

DEDICATION

To my father, who has always given me something to aspire to.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Michael Ball, for his guidance, for his belief in me and for introducing me to the realm of possibilities within integer programming.

Also, I thank the members of the Collaborative Decision Making working group for many hours of meetings and conversation that helped to educate me in air traffic management. In particular, I thank Michael Wambsganss, Roger Beatty, Rick Oiesen, and Jim Wetherly, each of whom has been inspirational in his leadership and insight in the future of air traffic management.

Also, I thank the members of my dissertation committee, Saul Gass, Paul Schonfeld, William Widhelm and David Lay, for their insightful comments and helpful suggestions.

This work was supported by the Federal Aviation Administration through the National Center of Excellence in Aviation Operations Research.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

This dissertation develops innovative techniques in the field of integer programming while providing practical solutions to some of today's most problematic areas of air traffic flow management (ATFM). Because ATFM problems are combinatorial in nature, they lend themselves nicely to modeling by integer programs. However, integer programming is a difficult field in its own right and the models developed here have raised issues of theoretical importance that transcend the original applications. These issues are resolved with both new and existing techniques of integer programming.

Two major problems of ATFM are solved by this dissertation. Each is an extension of the ground holding problem (GH) in which a decision maker at the FAA (Federal Aviation Administration) must determine the optimal amount of ground delay to assign to each of the flights bound for a common airport with limited arrival capacity.

In the first problem, the banking operations of the major airlines are considered. As it stands, there is no provision made by the FAA for holding banks of flights together in time. Several means are provided for the addition of banking constraints to an existing formulation of the ground holding problem.

The integer program used to solve the ground holding problem with banking constraints (GHB) serves as a prime example in which the addition of side constraints can transform an integer program from one that is readily solved to one that is computationally very difficult. It is becoming increasingly evident in the field of integer programming that tractability can be greatly enhanced by reformulation of constraints. This fact is dramatically accentuated by the empirical work of Chapter 3. The five formulations proposed run the full spectrum of computational performance varying, in some instances, by several orders of magnitude.

In any integer program, one seeks a formulation for which the solution set of its linear programming relaxation fits tightly around the convex hull of integer solutions. The closeness of fit is reflected in the value gap, meaning the difference between the function values of the integer program (IP) and the linear program (LP) relaxation. The strongest formulations are those that define facets of the convex hull of integer solutions.

The challenge of GHB is to write a facet-inducing model that yields integer solutions in real time using modest computing power and readily available IP solvers. This was achieved through the "ghost flight" model, XGF. XGF provides an efficient means by which to solve the underlying ATFM problem.

The outstanding performance of XGF is chiefly attributable to the establishment of auxiliary variables to create an extended formulation. The augmentation of the original problem with new variables runs counter to the mathematical "intuition" that the simplest formulation of a problem is the best. The combination of variable augmentation and constraint reformulation demonstrates the need in integer programming for creative formulations.

GHB can be generalized to a job-scheduling problem in which a batch of jobs are to be processed sequentially on a single machine subject to the constraint that certain categories of those jobs must be scheduled closely together. Therefore, the applications of XGF extend beyond ATFM.

In the second problem solved by this dissertation, the stochastic nature of arrival capacity in the ground holding problem is addressed. The FAA currently employs first-come, first-served allocation schemes to assign ground delay to each flight bound for an airport suffering from constricted arrival capacity. The optimality of these assignments is contingent upon advanced knowledge of the airport acceptance rate for each time period over the planning horizon. Unfortunately, these rates are rarely known with certainty in advance, since they are (in most cases) dependent upon weather. The stochastic ground holding problem seeks to quantify this uncertainty and to achieve a balance between deterministic ground holding and expected airborne delay.

In Chapter 4, a flow model is introduced to solve the stochastic ground holding problem. The greatest challenge in the design of this model was to avoid the complexities arising from an explicit computation of the length of time that each flight is held in each of the two states, ground holding and (expected) airborne holding. This difficulty was surmounted by treating flights on an aggregate level. In fact, the output of the model is simply the number of flights that should be released from a ground-holding state at each time period. It is shown that this is sufficient to solve the overall problem. Moreover, this can be extended to other stochastic production models in which goods can be held in two forms of inventory and a natural ordering is imposed on them.

Network flow problems are desirable for two reasons. First, they can be efficiently solved by known algorithms that are, in many cases, faster than the simplex method. Secondly, the constraint matrix is totally unimodular (TU), meaning that every square submatrix has determinant $0, 1$ or -1. Therefore, integer solutions can be obtained directly from the LP relaxation. Although the stochastic model presented in Chapter 4 is a flow model, it is shown not to be a (primal) network flow model because of the coupling constraints required at each time component. It can be directly shown, however, that the matrix is TU.

To date, every TU constraint matrix arising out of a natural application has proven to be a network or have a dual network. Having shown that the primal of the constraint matrix for the stochastic flow model is non-network, the possibility that the dual is a network is thoroughly explored. The positive findings give insight into the flow model and provide an alternative proof that the primal and dual matrices are TU.

In contrast to the variable augmentation technique performed on the banking constraint problem, the preprocessing technique applied to the stochastic flow model shows that the elimination of certain variables from a formulation can also expedite solution procedures. In this instance, a recursive procedure is used to fix a priori the values of those variables corresponding to flights exempted from delay.

Over the last two decades, the increase in air traffic at the major airports in the United States has vastly outgrown the increase in airport resources. Reduced arrival capacity at an airport is currently the leading source of air traffic delays. Inequities between capacity and demand are met by the FAA with ground delay programs (GDP's) in which inbound flights are held on the ground at their departure gates in lieu of costly and hazardous airborne holding. The equitable and efficient management of these GDP's has become a burden to the FAA and a bone of contention with the airlines.

The old paradigm in which the FAA acts as a centralized authority and decision maker is giving way to a new age of collaboration in which the primary users of the National Airspace System (NAS), the airlines, are given more control over their operations. In particular, they will work more closely with the FAA on those decisions that are more economic in nature and less concerned with safety. The collaborative decision making group is largely responsible for this shift to a more collaborative setting in air traffic management.

The improvement of ground delay programs run by the FAA was the original focus of the Collaborative Decision Making (CDM) working group but they are turning their efforts toward other aspects of air traffic management such as the routing or aircraft around severe weather conditions. CDM efforts are rapidly reshaping cultural behavior in ATFM and are bound to have a dramatic impact on ATFM and air traffic research. This dissertation presents the philosophies and methodologies of this influential group and provides a careful analysis of their heuristics for the allocation of arrival slots during a ground delay program.

Although CDM methodologies for ground delay programs will be implemented within the coming months at two major airports, they have not yet been officially embraced. The researcher in ATFM finds himself straddling two worlds: the current world, under the paradigm in which resources are allocated by a central decision maker, and the new world, in which decisions (not totally safety related) are made in a collaborative setting.

The banking constraint models presented in this dissertation assumes a central authority who tries make maximally efficient use of airport resources while minimizing the economic impact of a ground delay on the hubbing operation of an airline. These models are designed to be used in conjunction with the current paradigm of ATFM. The stochastic flow model presented in Chapter 4 is designed to be an integral part of the iterative cycle of decision-making currently being developed by CDM. Using the optimal acceptance rates output by the model, the FAA can make initial arrival slots assignments for incoming flights and which the airlines can then modify according to the ground rules they have established.

The remainder of this dissertation is organized as follows.

Chapter 1 (remainder): Background information on air traffic flow management is provided along with an encapsulation of the key issues in integer programming.

Chapter 2: This chapter describes ground-holding strategies and paradigms employed by the FAA for dealing with limited arrival capacities. These strategies are heuristic solutions to the static, deterministic ground-holding problem. A brief review of the literature on all versions (e.g., dynamic, multi-airport) of the ground-holding problem is provided. Also, Chapter 2 discusses the goals and current scope of CDM, the motivating forces behind its development, and the methodologies that have been developed by CDM.

Chapter 3: Both the single airport ground-holding problem (SAGHP) and the multi-airport ground-holding problem (MAGHP) can be extended by the addition of banking constraints to allow for the hubbing operations of major airlines. In this chapter, five different models and several variations of the (single-airport) ground holding problem with banking constraints are presented. The models are evaluated on both an analytic and empirical level.

Chapter 4: The main tool employed by the FAA for controlling flights bound for an airport with restricted arrival capacity is a Ground Delay Program (GDP). The effectiveness of a GDP is entirely dependent upon advanced knowledge of airport capacity and demand. Since the bulk of airport demand consists of scheduled flights, demand is generally predictable. Airport capacity, however, varies with meteorological conditions and is, therefore, highly stochastic in nature. In this chapter, an integer programming model of the stochastic ground-holding problem is presented. The model finds the optimal trade-off between airborne delay and ground delay in the formulation of a GDP. Proof is provided that the model yields the integer solution directly from its linear programming relaxation and the network structure of the dual of the linear programming relaxation is revealed. Using a commercial solver, six realistic test cases of the stochastic ground-holding problem are solved to optimality, each in a fraction of a second.

Chapter 5: The closing remarks cite the major contributions of the dissertation and point toward areas of future research.

Appendices: Appendix A contains a formal discussion of two algorithms presented in Chapter 2. Appendix B contains details of the proofs of lemmas and theorems found in Chapters 3 and 4. Appendix C contains computational results from Chapter 3. Appendix D contains a glossary of acronyms, mathematical terms and notation used throughout the dissertation.

1.1 Background: Air Traffic Flow Management

The airspace in the continental United States is partitioned into 22 sectors. Associated with each sector is an Air Route Traffic Control Center (ARTCC) operated by the FAA and a number of waypoints used for monitoring air traffic. Each ARTCC guides aircraft from waypoint to waypoint until they have passed into another sector or arrived within 200 miles of their respective destination airports, at which point control of the aircraft is assumed by the airport control tower. Air traffic for the entire airspace in the United States. is coordinated by the Federal Aviation Administration's (FAA) central control facility in Herndon, Virginia, known as the Air Traffic Control Systems Command Center (ATCSCC). The flight paths, control facilities, airports, and waypoints comprise the National Airspace System (NAS).

The problem of managing the safe and efficient flow of air traffic flow throughout the NAS is known as the (air) traffic flow management problem (TFMP). If the capacity for air traffic in the NAS were infinite, TFMP would be reduced to collision avoidance. However, in a given unit of time, there are limitations imposed upon the number of aircraft that can

- pass through a waypoint
- reside in a sector of airspace while maintaining separational distances
- be safely monitored by the crew of air traffic controllers on duty
- depart from an airport
- arrive at an airport

For a given component of the NAS, the **capacity** of the component is the number of aircraft that can be processed in a given unit of time, while **demand** is the number of aircraft that are scheduled (or predicted) to be processed in a given unit of time. Throughout this dissertation, the term **capacity-demand inequity** (CDI) refers to a situation in which capacity is below that of demand, the reverse case not being problematic. A CDI can occur at any one of three components in the NAS: a waypoint, terminal facility (airport), or in the airspace itself. The occurrence of a CDI within the airspace is rare - the airspace in the United States is large relative to the volume of traffic it generally supports. The occurrence of a CDI at a waypoint or terminal facility, however, is quite common.

A CDI at a waypoint is an unfortunate by-product of the monitoring paradigm used by the FAA. Aircraft are passed from waypoint to waypoint by the air traffic controllers, hence, tend to accumulate at those waypoints. Vast strides have been made in the tracking capabilities for both flight crews and air traffic controllers. The precise location of aircraft can now be maintained throughout flight duration by means of a Global Position System (GPS). Also, aircraft can detect each others presence long before (human) visual contact is possible. These technological innovations, though not yet implemented, are paving the way for **free flight**, in which planes would be allowed to fly the path of their choice between city pairs provided that it meets constraints disseminated by the FAA at the time of travel. A version of free flight is currently in effect for aircraft flying at altitudes of more than 22,000 feet.

Over 90% of the delays in air travel are caused by a CDI and subsequent congestion at one of only 22 of the 18,224 airports in the country [23]. Not surprisingly, these 22 locations coincide with the major metropolitan areas, which are prime destinations and points of origin for passengers. Air traffic demand is projected to grow. For these reasons, limited airport capacity is a major concern to the airlines, the FAA and the public.

One view to take on these CDI's is that there is simply too much demand at the major airports, particularly at morning and evening hours. This uneven distribution of demand is further aggravated by the fact that many of the commercial carriers operate a hub-and-spoke system in which flights tend to originate at or depart from a common location (hub). There would be little support, however, for any demand redistribution plan that asks the airlines to curtail well-established operational paradigms or that asks the general public to travel at odd times of the day to unpopular locations. For these reasons, it seems more promising to focus efforts on increasing airport arrival capacity rather than decreasing demand.

Long range plans to increase airport arrival capacity include the construction of new facilities and the expansion of existing ones. Unfortunately, these plans are impeded by the fact that real estate in major metropolitan areas is often unavailable or prohibitively expensive. Also, expansion of facilities is often blocked by surrounding communities, who are concerned with noise and automotive traffic congestion. Until such a time that there are sufficient resources to meet demand at all times of the day, there needs to be maximally efficient use of airport resources.

Insufficient airport capacity affects both arrivals and departures. In this dissertation, however, only instances of limited arrival capacity are addressed. Although limited departure capacity leads to passenger delay and added crew costs for the airlines, limited arrival capacity has both of these undesirables plus airborne holding. Airborne holding is more dangerous than ground holding, incurs extra fuel costs for the airlines, and increases the stress level for air traffic controllers.

1.2 Background: Integer Programming

An integer program (IP) is a problem in which one wishes to optimize a linear functional $f(x)$ subject to the conditions that (i) the vector x satisfies a system of linear inequalities $Ax \leq b$ (or $Ax \geq b$), (ii) each component of x be non-negative and (iii) each component of x be integer. If this last condition is relaxed, then one obtains a linear program (LP), which is called the LP relaxation of IP.

Consider the LP below.

$$\begin{gathered} min\ f(x) \qquad (1.1) \\ subject\ to \\ Ax \geq b \\ x \geq 0 \end{gathered}$$

Let P be the polyhedron defined by the system $Ax \geq b$, $x \geq 0$. Using the fact that P is convex, (1.1) can be solved very efficiently by algorithms such as the simplex method.

If the constraint that x must be integral is added to (1.1), then one obtains an IP with a set F of feasible points. F is a discrete set of lattice points within P but, unless F consists of a single point or no points at all, F is not convex. This renders useless many traditional optimization procedures. For this reason, integer programming is, in general, more difficult than linear programming.

In any practical application of an IP, one can find some upper bound on the integer values that can be assumed by the variables. Therefore, there is always an algorithm that will find the optimal value to the (bounded) IP in a finite number of steps: simply enumerate all the feasible points and select the one with the optimal function value. However, this is highly impractical for all but the smallest problems. Consider that a problem with just 100 binary variables could have 2^{100} feasible solutions.

The algorithms employed by commercial solvers for solving integer programs or mixed integer programs (in which both integer and non-integer variables appear) are based on a branch-and-bound (B&B) strategy. In B&B, the set of feasible points is enumerated implicitly by exploring the tree of solutions in which each branch represents a restriction of the set of feasible points. At each node, a solution is evaluated and a decision is made as to whether or not the optimal solution could lie in the subtree below the current node. The pruning criteria are based on the solution to the LP relaxation that is obtained after imposing the restrictions implied by the current path through the B&B tree. When minimizing, for instance, if the LP relaxation value is higher than the current best integer value, then the optimal solution could not possibly lie in the subtree and the subtree is pruned. In practice, the success of any B&B algorithm lies in its ability to do a great deal of pruning.

When solving an IP, the number of integer solutions that need to be explored can be greatly reduced by relaxing the integer constraint on some of the variables. In Chapter 3, an instance of an IP will be presented in which only a small portion of the variables need to be declared integer before solving the problem. In fact, these auxiliary variables were added to the problem specifically to expedite the solution process.

Often, much can be done to simplify an IP after it is formulated but before it is solved. It might be possible to fix some of the variables á priori or even eliminate variables altogether. Bounds can be examined for potential tightening and constraints can be examined for elimination. Any such steps fall under the category of **preprocessing**. Preprocessing plays a crucial role in the work in Chapter 4.

Every polytope (bounded polyhedron) P has a minimal description meaning that, up to multiplication by constants, there is a unique set of linear inequalities that describe it. Given an inequality $\pi x \geq b$(or $\pi x \leq b$), the set of points in P that satisfy π at equality is called a **face** of P. A **facet** of P is a face of dimension $n-1$, where n is the dimension of P. If π is one of the inequalities necessary to the description of P, then the face defined by π is a facet of P. Since the convex hull of integer points feasible to an IP is a polyhedron, it has a minimal description.

A major technique in integer programming is to produce a formulation in which as many as possible of the inequalities represent facets of the convex hull of integer solutions, P_C. In the best of circumstances, the solution to the IP can be obtained directly from its LP relaxation. This is possible whenever each of the facets of P_C is represented by at least one of the inequalities. Since a minimal description of P_C is rarely known, a greater number of constraints might be preferable to a lesser number so that P_{LP} is as close to P_C as possible; this increases the chances of solving the problem quickly.

Let P_{LP} be the set of points feasible to the LP relaxation of the integer program, IP. If an integer point is feasible to IP, then it is feasible to the LP relaxation and, therefore, P_C is contained in P_{LP}. If P_{LP} fits tightly around P_C, then the formulation at hand is said to be a **strong formulation**. Let IP' be an equivalent formulation of IP, meaning that the set of integer points feasible to IP' is the same as those integer points feasible to IP. Let P'_{LP} be the set of points feasible to the LP relaxation of IP'. If P'_{LP} is contained in P_{LP}, then $P_C \subseteq P'_{LP} \subseteq P_{LP}$ and IP' is said to be a stronger formulation than P_{LP}. IP' is the strongest formulation possible if $P_{LP} = P_C$.

It will be demonstrated in Chapter 3 that the ability to solve an integer program can vary enormously with the choice of the formulation. Therefore, it is important to compare both analytically and empirically a number of alternative formulations and choose the strongest one. A valuable experimental metric used for determining the stronger of two models is **value gap**, meaning the difference between the LP relaxation optimal function value and the optimal integer function value. A lower value gap generally indicates a stronger model.

In the event that each of the corner points of (non-empty) P_{LP} is integral, there will always be an integral optimal solution x^* to the LP relaxation. We call such a polyhedron an integral polyhedron. Every integer point feasible to IP is contained in P_{LP}. Since x^* yields the best function value of all points in P_{LP}, in particular, it yields the best function value of all integer points in P_{LP}, so x^* must be the optimal integer solution. The ideal formulation of an IP is one for which the LP relaxation is integral for, then, the IP can be solved by applying an LP optimization algorithm such as the simplex method to the LP relaxation of the IP.

Let $Ax \geq b$ be the linear system that describes the set P_{LP}. One way to ensure that P_{LP} is integral is to show that the matrix A is totally unimodular (TU), meaning that every square submatrix of A has determinant 0, 1, or -1. In Chapter 4, a formulation of the Stochastic Ground-Holding Problem will be presented which yields a TU matrix. Moreover, the dual of the LP relaxation of this formulation can be transformed into a network flow problem and solved by specialized network algorithms that are, in many case, faster than the simplex procedure.

In this dissertation, the aforementioned techniques of integer programming are applied toward the formulations of problems in air traffic flow management.

Chapter 2

New and Existing Strategies for Ground-Holding

2.1 The Ground-Holding Problem

The Air Traffic Control Systems Command Center (ATCSCC) monitors airports throughout the United States for capacity-demand inequities. Whenever it is predicted that the number of flights arriving at an airport within a 15-minute time interval will exceed the number of flights scheduled to land, the ATCSCC is required by FAA regulation to take some form of action.[1] Short-term periods of capacity-demand inequities are alleviated by airborne tactics such as re-routing and variations in airborne speed. Longer-term periods of capacity-demand inequities are met by the ATCSCC with ground-holding strategies in which aircraft are held at their departure gates in lieu of costly and dangerous airborne delay.

In some cases, the ATCSCC will issue a ground stop in which all flights incoming to an afflicted airport are held on the ground at their departure gates until

[1]Much of the information in this section concerning the opertation of the ATCSCC was obtained through meetings with ATCSCC personnel.

airport arrival capacity rises above demand. These ground stops are reserved for extreme cases in which arrival capacity was severely underestimated or dropped suddenly without warning.

The primary tool of the ATCSCC for addressing arrival capacity-demand inequities is a ground delay program (GDP). In a GDP, each flight scheduled to arrive at an afflicted airport over a fixed time period is held at its departure gate long enough to ensure that it will be able to land without delay. For instance, if flight f is due to arrive at airport A at 12:00 and it is known that f will not be able to land until 12:30 due to limited arrival capacity at A, then f would be held at its departure gate for 30 minutes. The construction of a GDP requires the assignment of both a controlled time of departure (CTD) and a controlled time of arrival (CTA) to each incoming flight . Since en route travel times can be predicted with reasonable accuracy, the CTD of each flight is easily calculated once its CTA is known. The CTA is set once the flight has been assigned an arrival slot.

Currently, the ATCSCC assigns arrival slots by a first-come, first-served algorithm known as **Grover Jack**. A list of incoming flights is formed, ordered by the most recent estimated time of arrival (ETA) of each flight. The time horizon for the GDP is divided into hourly periods $t = 1, 2, ..., T$. Usually, $T = 4$. The arrival acceptance rate (AAR) of a time period t, denoted X_t, is defined as the number of aircraft that can be accepted during t. (Strictly speaking, this is not a rate. None the less, it is the established terminology in the air traffic community.) A value of X_t is set for each time period t by a specialist at the ATCSCC. In spirit, at least, the Grover Jack algorithm assigns controlled arrival times as follows. The first X_1 flights on the list are assigned to the first time period, the next X_2 flights are assigned to the second time period, and so on, preserving order of the list. The net effect of the Grover Jack algorithm is to stretch out the list of incoming flights over time.

In actuality, there are several complications that need to be addressed in the Grover Jack algorithm. Since a flight cannot be assigned to a time slot earlier than its ETA, some time slots will be passed over during the assignment process and have no flight assigned to them. International flights, general aviation, and flights airborne at the time of formulation of a GDP are exempt from the program, meaning that they cannot be issued a ground delay. In addition, the specialist may choose to exclude other categories of flights from ground delay, usually based on geographical location of point of origination. The arrival of these flights must be taken into account when assigning CTA's to flights.

Although the airlines agree that a first-come, first-served algorithm is an equitable method for distribution of arrival slots, they object to the use of ETA as the criterion for 'first-come'. They have cited the following scenario, known as the **double penalty issue**. Suppose that flight f is scheduled to arrive at airport A at 10:00 hours. If f is delayed for 30 minutes with, say, mechanical failures, then the ETA of f would be updated to 10:30. If a GDP is implemented in which f is assigned a 30 minute delay, then f would be given a CTA of 11:00. Overall, f has suffered a total delay $30 + 30 = 60$ minutes. The airlines feel that f has been penalized twice: the 30-minute mechanical delay, they feel, should have served as the program delay for f and f should have been assigned a CTA of 10:30, not 11:00. In essence, the airlines argue that the original scheduled time of arrival, not ETA, should dictate landing order in a GDP.

No matter what criterion is used for first-come in the formulation of a GDP, it is highly unlikely that any simple ordering of flights will be preserved when the program is finalized. One reason, already mentioned, is that a significant portion of the flights are not subject to the ordering criteria because they are exempt or excluded from the program. Also, flights can be delayed (or cancelled) by the airlines for reasons such as mechanical difficulties.

The problem of assigning ground delay to flights bound for a single airport can be mathematically modeled as an assignment problem known as the **ground-holding problem** (GH). The model requires the following assumptions.

Assumption 1: (Discrete time horizon) There is a fixed time horizon which has been discretized into T equally-sized contiguous time periods, $t = 1, 2, ..., T$.

Assumption 2: (Deterministic demand) The number of incoming flights is known in advance; for each flight f, there is a scheduled time (period) of arrival, denoted a_f (this is the earliest arrival time that can be assigned to the flight).

Assumption 3: (Deterministic capacity) For each time period, t, let b_t be the arrival acceptance rate (AAR) of the airport, meaning the maximum number of flights that can be accepted by the airport during that time interval. Then we assume that b_t is known in advance for each time period t. Strictly speaking, this does not hold in practice because the AAR's are dependent upon weather conditions and runway configurations, which are stochastic in nature. However, the specialist who formulates the GDP fixes these numbers in accordance with the current best estimate, so, for purposes of this formulation, we will assume that they are deterministic and known in advance.

Let F be the set of incoming flights that require arrival slots. We define for each f and each t, a binary variable, $X_{f\,t}$, such that

$$X_{f\,t} = \begin{cases} 1, & \textit{if flight f is assigned to time interval t} \\ 0, & \textit{otherwise.} \end{cases}$$

Each flight must be assigned to exactly one time interval so for each f we have one constraint of the form $\sum_{t=a_f}^{T} X_{f\,t} = 1$. The number of flights that are assigned a CTA within time period t cannot exceed the capacity b_t, so for each time interval t there is one capacity constraint of the form, $\sum_f X_{f\,t} \leq b_t$.

Let C_f be the cost of delaying flight f for one time period and let $\sigma > 1$ be a fixed parameter. Then the expression $X_{f\,t}\ C_f\,(t - a_f)^{\sigma}$ represents the cost of assigning flight f to time t and the summation of this term over all t and all f is the total delay cost. The parameter σ yields super-linear growth in the tardiness of a flight as t increases. This favors the assignment of a moderate amount of delay to each of two flights rather than the assignment of a small amount of delay to one and a large amount to the other.

In all, we have the following integer program.

(GH)

$$Minimize\ \sum_{f \in F} \sum_{t=1}^{T} X_{f\,t} C_f\,(t - a_f)^{\sigma} \tag{2.1}$$

$$subject\ to$$

$$\sum_{t=a_f}^{T} X_{f\,t} = 1 \tag{2.2}$$

$$\sum_{f} X_{f\,t} \leq b_t \tag{2.3}$$

$$0 \leq X_{f\,t} \leq 1 \tag{2.4}$$

$$X_{f\,t} \in \{0, 1\} \tag{2.5}$$

Let GH$_{LP}$ be the LP relaxation of GH, that is, the problem that results from relaxing constraint set (2.5). GH$_{LP}$ is a transportation problem. Since it can be shown that there is an integer optimal solution to every transportation problem, LP solvers or specialized transportation codes can be applied to GH$_{LP}$ to rapidly obtain the (integer) solution to GH.

GH was first systematically described by Odoni in [18]. Andreatta and Romanin-Jacur [3] treated the stochastic version of GH for an airport with constrained arrival capacity in (at most) one time period. In [25], Terrab and Odoni developed a dynamic programming formulation for the stochastic version of GH as well as heuristics to handle the larger cases. Using stochastic linear programming with recourse, Richetta and Odoni expanded this work to include the dynamic case, in which ground-holdings are updated as time progresses (see [21]). Although their dynamic solution yielded considerable savings over the static solution, the speed of solution proved to be too slow for realistic cases. See [7].

Ideally, the ground-holding problem should be solved on a network-wide level, taking into account the connectivity of flights. Flights can be connected in one of three ways: by passenger, crew or aircraft. In the former sense, passengers are scheduled to travel from airport A to C by taking a flight from A to B, then B to C. The arrival of the first flight should coincide (roughly) with the departure of the second flight. In the latter two senses, a single crew or aircraft may be scheduled to traverse many flight legs, e.g., from city A to city B to city C, and so on. The delay of even a single flight can propagate throughout the entire system.

Both GH and air traffic flow management in general have been treated on a network-wide level (taking multiple airports and flight connectivity into account) in Attwool [5], Sokkapia [24], Andreatta and Romanin-Jacur [3], Wang [32] and by Vranas, et. al., in [29] and [30], and, more recently, by Bertsimas and Stock [8].

In practice, there are several problems associated with applying network models of air traffic flow. The first is that the models are generally difficult (NP-hard), integer programs. Only greatly simplified versions have been solved in real time. Secondly, the models require extensive, timely information on all flights in the system. The airlines are not prepared and not always willing to supply such information. Nor is there presently a communications system capable of handling all the information. Third, and most importantly, the airlines and the FAA are moving away from the type of centralized control of air traffic that is assumed by such a model. Only single-airport scenarios are considered in this dissertation.

2.2 Collaborative Decision Making in Air Traffic Flow Management

2.2.1 A Time for Change

The implementation of a GDP is generally met with trepidation by the airlines because it represents government intervention and exposes airline operations to inaccurate estimations on the part of the ATCSCC. For instance, if the future arrival acceptance rate of an airport is overestimated, then arrival demand will exceed capacity and planes will absorb delay in costly airborne holding patterns rather than on the ground at their departure gates. On the other hand, if future AAR's are underestimated, then arrival capacity will exceed demand, arrival slots become a wasted resource, and flights absorb unnecessary ground delay. The airlines find it particularly aggravating when a GDP is aborted in mid-operation. This happens whenever the ATCSCC has clear evidence that the original capacity and weather forecasts were overly pessimistic. Estimates vary between the airlines and the FAA as to the percentage of GDP's that are aborted, but these estimates are as high as 60%.

To a large degree, a GDP is based upon the economic premise that ground-holding is cheaper than airborne-holding. While this is certainly true on average, it does not hold for every aircraft in every delay situation. Certainly, the airlines recognize the need for centralized control of air traffic into an airport with limited arrival capacity but these situations serve as a prime example of a major objection of the airlines to the current procedures. The objection is that the FAA is making economic decisions on behalf of the airlines. These are generally well intending but it has been well established that the FAA does not have the expertise or timely data on daily airline operations for effective decision making.

The responsibility of the FAA is to maintain the safety of the users of the NAS and to manage air traffic flow within the NAS in a manner that makes efficient and equitable use of resources. The current structure imposed on the system stems from the belief that the FAA and its agencies should act as a centralized authority and decision maker. This paradigm, as well as the procedures and standards for air traffic management, were developed just after World War II. Despite the enormous growth in air traffic in the United States in the last fifty years, this paradigm remains in effect today, largely unaltered.

A joint effort between government agencies and the airline industry known as Collaborative Decision Making (CDM) has arisen. The driving philosophy behind CDM is that improved data exchange and communication between aviation transportation organizations will lead to better decision making in air traffic flow management and that, whenever practical, those decisions which have potential economic impact on airline operations should be decentralized and made in collaboration with the airlines

The roots of CDM can be traced back to informal meetings of airline representatives in 1992 who were concerned with Ground Delay Programs. In the summer of 1993, the FAA program called FAA-airline data exchange (FADE) began. FADE was a short-term experiment to determine if air traffic management decisions would be affected by schedule updates from the airlines. The findings were positive. In the mean time, the Mitre Corporation was commissioned by the Air Traffic Management Integrated Products Team division of the FAA to find alternative processes by which airlines substituted flights into time slots filled by their own flights. The positive findings of the FADE program and the Mitre Corporation merged to form the two main components of CDM, data exchange and the development of tools and methodologies to improve air traffic management decisions.

As CDM has grown, so has its participation. The passenger airlines and FAA have been joined by the package carriers such as Federal Express and United Parcel Service (UPS), who have a vested interest in the outcomes of new methodologies being established by CDM. The technical challenges posed by CDM have solicited the expertise of a scientific consulting firm, Metron Inc., Mitre Corporation and a host of contractors. In 1995, Congress created a National Center of Excellence for Aviation Operations Research (NEXTOR) which centers around a consortium of four universities, one of those being the University of Maryland. (See [27] for a reference on NEXTOR.) Representatives from the CDM participating organizations, which now includes NEXTOR, gather monthly for working-group meetings. Many of the airlines have devoted full time representatives to CDM.

A novel relationship between the scheduled carriers and the FAA was formalized in the document, "Roles and Responsibilities", written by the FADE program manager and airline representatives in early 1995. It specifies that the ATCSCC should remain a neutral party and act as a service provider to the users of the NAS. The responsibility of the ATCSCC is to alert the users to situations within the NAS that place constraints upon their operations. The users are responsible for responding to those constraints with actions, intents and preferences that lie within those constraints. At least with respect to GDP's, the efforts of CDM have pushed air traffic management in the direction of decentralized decision making, thus giving the scheduled carriers input to air traffic management and greater control over their operations.

2.2.2 CDM Methodologies and Tools (appended by Appendix A)

The primary focus of CDM has been ground delay enhancements and the improvement of the cancellation and substitution process used by the airlines. At the time of this writing, CDM methodologies have not been put into place. Prototype operations involving ground delay program enhancements are scheduled to go in effect at San Francisco and Newark airports in early 1998. CDM efforts are extending to any constrained situation in the NAS. Given the current momentum of CDM and its growing support, it seems inevitable that CDM methodologies will have a major impact on air traffic flow management, particularly in the context of ground-holding strategies. It is appropriate that the current body of aviation research should reflect, if not be tailored to, the CDM philosophy. In this section, we examine the procedures and technologies that have arisen out of CDM efforts.

The ubiquitous operational procedure for handling a constrained situation in the NAS is a cycle of feedback between the service provider and the users of the NAS. For example, suppose that the service provider (ATCSCC) announces that at JFK airport the arrival acceptance rate will drop from 50 to 30 flights per hour over a two-hour time period and that the scheduled demand (of incoming flights) over those two hours is 45 flights per hour. A GDP is issued and each non-exempted flight bound for JFK is issued a controlled time of arrival (CTA) and a controlled time of departure (CTD). This throws the users (airlines) into a state of irregular operations and internal compensations must be made. The users are then given the opportunity to respond with flight cancellations, substitutions or diversions.

In a typical round of cancellation/substitution, each airline is given a list of their flights. Associated with each flight is an arrival slot, assigned under the GDP. Each airline is said to own its slots and can redistribute its flights over those slots. For instance, suppose that flight AL100 is a lightly loaded flight with few connecting passengers and a CTA (controlled time of arrival) of 12:00 and that AL500 is a fully loaded flight with many connecting passengers and a

CTA of 12:45. Since the timely arrival of the cargo plane is not so crucial, airline AL might want to cancel (or divert) AL100 and substitute AL500 into the 12:00 time slot, thus saving AL500 45 minutes of delay. This opens the 12:45 time slot and AL can consider other flights for substitution into that time slot. Another allowable operation is the exchange of time slots between two flights so that one is moved earlier in time and the other is moved later. This is useful whenever the minimization of delay of one of the flights is paramount. See [31] for a more detailed treatment of cancellations and substitutions.

Next, the service provider processes the information provided by the users (the new arrival times), and revises the forecasts on arrival capacities and demands. A new set of operational constraints (in this case, revised CTA's) is issued. The users are again given the opportunity to respond, and the cycle repeats, probably in time blocks of 30-60 minutes per cycle.

The overall procedure assumes a constrained situation that can be anticipated an hour or more in advance, such as reduced airport capacities or re-routing around severe weather patterns. Fortunately, most CDI's (capacity-demand inequities) in the NAS arise from bad weather and fall into this category.

Note that the service provider can make an accurate situational assessment at each iteration of a data exchange cycle only if the airlines are supplying updated data in the form of cancellations, revised ETA's, and so on. Up until now, there has been little incentive for the airlines to do so. In fact, one of the impediments to effective air traffic management has been the reluctance of the airlines to supply data to the FAA. Their concern has been that information would be (inadvertently) used against them, as in the double penalty issue (see section 2.1).

The incentives for the airlines to provide timely data to the FAA during the formulation of a GDP have been housed in the two algorithms, **ration-by-schedule** *(***RBS***)* and **compression**. We now give brief descriptions of those algorithms and their purpose. More detailed analyses and formal presentations can be found in Appendix A.

The RBS algorithm: The purpose of this algorithm is to ration arrival slots according to original scheduled arrival times of flights and to serve as an initial assignment of CTA's for subsequent rounds of collaboration between the airlines and the FAA. The original scheduled arrival time of a flight is determined by the original gate time of arrival (OGTA) minus a standard ten-minute taxi time.

The primary difference between RBS and the Grover Jack algorithm currently used by the FAA is that RBS uses OGTA−Taxi time as the ordering criterion whereas Grover Jack uses ETA. Otherwise, the algorithms are very similar. Details aside, virtual arrival slots are created over a time horizon based on AAR predictions and the first flight is assigned to the first time slot, the second flight is assigned to the second time slot and so on. Under the RBS algorithm, airlines can effectively reserve slots at an airport by scheduling flights weeks ahead of time in the OAG (official airline guide). This removes the fear that of forfeiting an arrival slot by reporting a delay, as in the double penalty issue.

The initial assignment of CTA's made by RBS plays an important role in the subsequent cancellation/substitution process. Suppose that flight $AL100$ belonging to airline AL has been assigned to the t^{th} time slot. Then AL is said to "own" slot t. During a round of cancellation/substitution, AL is free to move another one of their flights, say $AL200$, from a less desirable time slot, $(t + k)$, into t, provided that t is a feasible arrival time for $AL200$. Flight $AL100$ will have to be cancelled, of course, or moved down into slot $(t + k)$, because the total number of slots owned by AL is fixed by RBS. By this process, each airline has the opportunity to minimize the damages of flight delays in a GDP.

The Compression algorithm: After a round of cancellation/substitution, the total number of flights in the program has probably been reduced through cancellations. This creates "holes" in the assignment schedule, meaning there are valid arrival slots with no flights assigned to them. The purpose of the compression algorithm is to move flights up in the schedule (earlier in time) to fill these slots.

There are two ways for an arrival slot to become empty. Either (1) the airline assigned control/ownership of the slot by RBS has simply declined to substitute a flight into the slot or (2) the slot is too early for any flight to be assigned to it by the controlling airline . In either event, the controlling airline would **release** the time slot to the compression algorithm which tries to fill the slot with another flight of the controlling airline or to make an appropriate compensation.

The abbreviated version of the compression algorithm is as follows.

1. For increasing values of t, the status of time slot t is checked. If t is filled, the algorithm moves on to slot $t + 1$.

2. If t is released, then it has a controlling airline, AL. The algorithm first scans down a list of flights from AL for a feasible assignment. If no AL flight is found, it searches for a feasible assignment on a list of all flights not in AL. If a flight f from AL (or another airline) is found for t, then f has been moved up from its current slot t'. Then, t' is declared released with controlling airline AL and the algorithm is applied to t'. This creates a stream of substitutions (possibly none) of flights upward through the

schedule. Eventually, this stream stops (see the full algorithm in Appendix A for stopping criteria) and the algorithm resumes at slot $t + 1$, where it left off.

The algorithm terminates when every slot has been filled with a flight or declared unusable.

The important features of the compression algorithm are that (i) arrival slots are filled whenever possible, (ii) flights from the controlling airline of a slot t are considered before all others when t is released, (iii) if the controlling airline cannot use a slot it is (eventually) compensated since it receives control of the slot vacated by the flight which moves into its slot and (iv) there is no way for an airline to involuntarily lose a slot reserved by RBS (provided that the criteria for the termination of a substitution stream are properly set). Issues such as the ordering of lists and the criteria for feasibility of a flight to be moved to a time slot are discussed in Appendix A.

In order to coordinate all of the activities surrounding the cancellation/substitution process and the foregoing algorithms, the participating airline operational centers (AOC's) and the ATCSCC have been networked by the newly established "AOC Net". The AOC Net carries the airline data to the ATCSCC and an aggregated demand list to each airline .

Collaborative decision making and common situational awareness of airport arrivals is made possible by a decision support tool called the **Flight Schedule Monitor (FSM)**. The major features of FSM are listed below.

- Common situation awareness: When used on-line, FSM displays to both the NAS user and the ATCSCC the latest projections of airport arrival demand at the user-selected airport. The demand is displayed in a time line fashion in which each tick mark is a minute of time and incoming flights are represented by icons on the time line. The position of the icon corresponds to the most recent ETA of the flight. Airlines can be distinguished by an icon coloring scheme.

- Analytical tools and strategy evaluations: To explore alternative ground-holding strategies, both the NAS user and the ATCSCC can formulate hypothetical ground delay programs to generate delay statistics and measure equity amongst the airlines. Also, FSM has an (off-line) historical mode in which the user can conduct analyses on archived flight data.

- Information processing: FSM (1) executes both the RBS and Compression algorithms, (2) allows users to make cancellations and substitutions and (3) processes an updated demand list together with user-specified parameters such as AAR's to generate a complete ground-holding strategy (such as GDP or ground stop). In particular, controlled times of arrival and departure are output for each flight in the program. However, only the ATCSCC has the capability to institute a program.

- Compliance monitoring: FSM allows the NAS users and service provider (ATCSCC) to verify compliance with established rules of conduct within the system. For instance, an alarm is tripped whenever a flight departs that was reported cancelled by an airline.

Although FSM was developed as the primary flow tool for the FAA, the airlines have made significant contributions and improvements to FSM through experimental sessions (war games), user feedback and concept design. FSM is in the final stages of testing and development and is now operating on-line at the ATCSCC and 14 airlines.

One can see that, from beginning to end, the CDM process for addressing degraded arrival capacity at an airport is rather involved. It promises to be an improvement, however, on the current system in which the ATCSCC acts as sole decision maker operating with incomplete information. By making the users aware of their own role in a constrained situation, it is possible that the need for government intervention can be greatly diminished or eliminated altogether.

CDM practices are just on the verge of implementation. For the remaining two chapters of this dissertation, we diverge into two perspectives. In Chapter 3, the addition of banking constraints to the ground-holding problem is considered, but under the old system, prior to CDM methodologies. In Chapter 4, the stochastic nature of airport arrival capacity is addressed in a manner compatible with CDM.

Chapter 3

The Addition of Banking Constraints to the Ground-Holding Problem

3.1 Hubbing Operations

Each major airline in the United States has chosen at least one airport as a hub of its operation. The hub acts as a base of operation and a central point of transfer for passengers, thus simplifying the enormous scheduling problem that confronts the airline. From an aerial view, the pattern formed by the flight paths resembles the spokes of a wheel with the hub at the center, hence, this type of operation has been dubbed “hub-and-spoke”.

The hub-and-spoke system allows an airline to pool at a central location those passengers with geographically diverse points of origin but a common destination (or the reverse). For instance, some of the passengers from flights A, B and C can be scheduled to transfer at the hub to a flight D with a destination common to all of them. But in order for this to work, the arrival of flights A, B and C need to be coordinated with the departure of D. Flights A, B and C form what is known as a **bank**, meaning, a group of flights whose arrival times must fall within a specified time window.

In the solution to the ground-holding problem (GH), the assigned arrival times of the flights tend to spread out over time because the number of flights that can be accepted per time period is less than in the original schedule. For instance, if 240 flights were scheduled to land over a four-hour time period and the arrival capacity of the airport were cut in half, then it would take eight hours to land those 240 flights. This tends to spread out the arrival of flights within a bank as well, often beyond an acceptable level.

Banking constraints can be added to the formulation of the GH to keep the flights of each bank temporally grouped. For each bank b, let Φ_b be the set of flights in bank b and let w_b be the width of b, meaning the maximum number of time intervals over which the flights of bank b are allowed to land. Note that the difference between the sums $\sum_{t=1}^{T} tX_{ft}$ and $\sum_{t=1}^{T} tX_{gt}$ is the difference between the arrival times of the flights f and g. Then the following constraint set, for instance, will ensure that the flights of b land in a time window of desired length.

Formulation 1: XTC (the time coefficient model)

$$\sum_{t=1}^{T} tX_{ft} - \sum_{t=1}^{T} tX_{gt} \leq w_b \;\; for\, all\, b, \;\; for\, all\; (f,g) \in \Phi_b \times \Phi_b \tag{3.1}$$

By adding (3.1) to GH, we have a model (XTC) of the ground holding problem with banking constraints (GHB). Unlike GH, the LP relaxation of the GHB rarely yields optimal integer solutions.

The ease with which an integer program is solved can vary dramatically with the formulation, so, it is important to find the best formulation possible. Often times, the most direct approach is not the best approach. Two formulations of an integer programming problem are said to be **equivalent** if and only if they have the same set of feasible solutions. Equivalent formulations can be derived by reformulating the constraints, selecting new variables, or augmenting the existing ones.

The purpose of this chapter is to find a strong formulation of GHB that can be solved rapidly using a commercial solver. In sections 3.2 and 3.3, several models of GHB are derived and the intuition behind them is explained. In section 3.4, the polyhedra induced by some of the more promising models are analyzed and, in section 3.5, we test the computational performance of each model on both real and artificially constructed data sets.

3.2 Alternate Models of GHB

Formulation 2: XW (the Window model)

Intuitively, it seems that the solving of GHB would be greatly facilitated by advanced knowledge of the time window in which each bank will arrive in an optimal solution. Each such window can be uniquely identified by its first time interval (i.e., the one with the lowest index value, t). This is the earliest time interval to which any of the flights of bank b can be assigned. So, for each bank, b, we establish a set of binary "marker" variables as follows.

$$Z_t^b = \begin{cases} 1, & if\, t\, is\, the\, first\, time\, interval\, open\, to\, bank\, b \\ 0, & otherwise. \end{cases}$$

The marker variables can be used to write a constraint that says, "if t is the earliest time interval open to the flights of bank b, then the arrival time of flight f in bank b must be no later than w_b units after t". We need one such constraint for each flight in each bank.

$$Z_t^b - \sum_{s=t}^{t+w_b-1} X_{fs} \leq 0 \quad for\, all\, t,\ for\, all\, b,\ for\, all\, f \in \Phi_b \tag{3.2}$$

The following set of assignment constraints ensures that the first time interval open to each bank is unique.

$$\sum_{t=1}^{T} Z_t^b = 1 \ \ for\, all\, b \tag{3.3}$$

The model XW is obtained by adding constraint sets (3.2) and (3.3) to GH. This model yields at most one banking constraint of type (3.2) for each pair (f, t), where $f \in F$ and $t \in \{1, 2, \dots, T\}$, and one banking constraint of type (3.3) for each b. Thus, the total number of banking constraints is $O(nT)$, where n is the number of bank flights and T is the number of time intervals.

Formulation 3: XMM (the Monotone Markers model)

An alternate formulation of the window constraint (3.2) can be written by directly translating the statement "if flight f (in bank b) arrives in time interval t, then one of the w_b intervals prior to t must be marked as the first interval open to bank b". This is the converse of the statement that generated (3.2) in the model XW. Rather than mark the first time interval by $Z_t^b = 1$, and $Z_t^b = 0$ for all other time intervals (as in XW), we choose to mark all time intervals strictly preceding the start of the window by the assignment $Z_t^b = 1$ and all subsequent intervals by $Z_t^b = 0$. (In essence, we are transforming the marker variables into Bertsimas-Stock variables - see section 3 for an explanation of these variables and why they might help).

$$X_{ft} - \sum_{s=t-w_b}^{t-1} Z_s^b \leq 0 \quad for\, all\, t,\ for\, all\, b,\ for\, all\, f \in \Phi_b \tag{3.4}$$

Constraint set (3.5) precludes the possibility that both $X_{f\,t} = 1$ and $Z_t^b = 1$ for a fixed t while constraint set (3.6) forces the marker variables to be monotonically non-increasing.

$$Z_t^b + X_{ft} \leq 1 \quad for\, all\, t,\;\; for\, all\, b,\; for\, all\, f \in \Phi_b \tag{3.5}$$

$$Z_t^b - Z_{t-1}^b \leq 0 \quad for\, all\, t,\; for\, all\, b \tag{3.6}$$

The model XMM is obtained by adding (3.4), (3.5) and (3.6) to GH. The number of banking constraints increases quadratically with the size of the problem and has an asymptotic bound of $O(nT)$.

Formulation 4: XSS (the Double Sum model)

The following simple constraint states that if flight f arrives in time interval t, then flight g cannot arrive in time interval s and vice-versa.

$$X_{ft} + X_{g\,s} \leq 1 \tag{3.7}$$

If we write one constraint of type (3.7) for each pair of bank flights f and g and for each pair of time intervals t and s such that $|t - s| > w_b$, then all the flights of bank b must arrive within a window of w_b units.

We can write a stronger version of (3.7), which states that if f lands in time interval t or earlier, then g cannot land in time interval $t + w_b$ or later, as below.

$$\sum_{s=1}^{t} X_{f\,s} + \sum_{ss=t+w_b}^{T} X_{g\,ss} \leq 1 \qquad for\, all\; t,\;\; for\, all\; (f,g) \in \Phi_b \times \Phi_b \tag{3.8}$$

The final model, XSS, is obtained by adding (3.8) to GH. We extend the notion of “arrival” to fractional solutions by saying that if $X_{f\,t} > 0$, then f has partially arrived at time t and f has fully arrived at the earliest time interval t for which $\sum_{s=1}^{t} X_{fs} = 1$. For each bank b, let $A_b = \{X_{ft} : \; f \in b\}$. Then in any solution to the linear relaxation of the GHB, one can compute the minimum and maximum values of t for which at least one of the variables in A_b is non-zero. We define the range of the bank in a given solution to be the difference of those numbers.

The strength of our latest formulation, XSS, lies in its ability to keep this bank range as small as possible in the LP. As an example of a fractional solution that is feasible to constraints of the type (2.2), (2.3) and (3.5) but not to (3.8), consider two flights, f and g, in bank b, with a specified bank width of $w_b = 2$ time intervals. Below are feasible assignments for the variables $X_{f\,t}$ and $X_{g\,t}$ for $t = 1, 2, \ldots, 8$.

$t =$	1	2	3	4	5	6	7	8
$X_{f\,t} =$	1/2	1/2	0	0	0	0	0	0
$X_{g\,t} =$	0	0	1/2	0	0	0	1/2	0

The model XSS has the undesirable feature that it produces a tremendous number of constraints for large problems. In fact, the number grows cubically with the size of the problem; it's asymptotic behavior is $O(n^2T)$. On the largest data set that we tested, 114,855 of the 115,174 constraints (i.e., 99.73%) were banking constraints. For problems of this size, even the compilation time of the C-program that writes the input for the solver CPLEX is significant: on the order of ten minutes. We now search for a model of equal strength that brings with it fewer constraints.

Formulation 5: XGF (the Ghost Flight model)

So far in our formulations of banking constraints, we have made pair-wise comparisons of the arrival times of the flights within a bank. But if we knew that, in every feasible solution to GHB, a "pilot" flight in the bank were going to arrive before the other flights in the bank, then we could compare the arrival of each bank flight to the pilot flight and cut down on the number of constraints by an order of magnitude.

There is no reason to believe, á priori, that every bank would naturally contain a pilot flight but we can add a ghost flight to each bank and write a constraint to enforce the arrival of the ghost flight before the other flights in the bank. For each bank b, we define a set of assignment variables, $\{Z_t^b : t = 1, 2, \ldots, T\}$, to mark the (fictitious) arrival of the ghost flight. That is, $Z_t^b = 1$ if the ghost flight arrives at time t and $Z_t^b = 0$, otherwise. The following constraint set will ensure that the arrival of each ghost flight is unique.

$$\sum_{t=1}^{T} Z_t^b = 1 \quad for\, all\, b \tag{3.9}$$

For each flight f in bank b, we write a constraint of the type (3.10) to ensure that the ghost flight will arrive before flight f and a constraint of type (3.11) to prevent the flights of each bank b from arriving more than w_b units behind the bank's ghost flight.

$$\sum_{s=t}^{T} Z_s^b - \sum_{s=t}^{T} X_{fs} \leq 0 \quad for\, all\, t,\ for\, all\, b,\ for\, all\, f \in \Phi_b \tag{3.10}$$

$$\sum_{s=1}^{t} Z_s^b + \sum_{ss=t+w_b}^{T} X_{f\,ss} \leq 1 \qquad for\ all\ t,\ for\ all\, f \in \Phi_b \tag{3.11}$$

The final model, XGF, is obtained by adding (3.9), (3.10) and (3.11) to GH. For every bank flight f and every time interval t, this model yields one banking constraint of the type (3.11) and one of the type (3.10). For every bank b and every time interval t, there is one constraint of the type (3.9). Thus, the total number of banking constraints produced by this model is $O(nT)$, where n is the number of bank flights. Contrast this with $O(n^2T)$ for model XSS.

In section 4, we will show that XSS and XGF are of equal strength, meaning that the optimal function value for the LP is the same for each model. Moreover, we will see that for both XSS and XGF, every banking constraint is a facet of the polyhedron formed by the set of integer solutions. This is most desirable because it greatly increases the chances of yielding an integer solution directly from the LP relaxation.

3.3 Variations on the Models

3.3.1 A Branching Technique

Recall that several of the formulations employ marker (Z) variables. If the (binary) value of each marker variables is fixed, then each banking constraint reduces to a trivial statement or is redundant to a non-banking constraint. The subsequent LP relaxation is a transportation problem and will yield an integer solution. Thus, we obtain a valid formulation by restricting only the Z variables to be integer. The IP solvers will then branch only on the Z variables.

This branching technique was applied to models XW, XMM, XSS and XGF. In Tables C.1-C.7, Appendix C, the reader will find rows marked "XWZ" and "XMMZ". These formulations are MIP (mixed integer programs) versions of XW and XMM, respectively, in which the integer constraints on the assignment variables (X_{ft}) have been relaxed. Neither XSS nor XGF model names are suffixed with a "Z" because these models were always solved with these relaxations. In section 5, we will analyze the benefits.

3.3.2 Bertsimas-Stock variables: A linear transformation

The standard assignment variables can be replaced by Bertsimas-Stock (B-S) variables, defined as follows.

$$W_{ft} = \begin{cases} 1, & if\, flight\, f\, arrives\, by\, time\, t \\ 0, & otherwise. \end{cases}$$

The assignment variables are defined so that for exactly one t, $X_{f\,t} = 1$. In contrast, the B-S variables are defined so that for every s greater than some t, $W_{f\,s} = 1$. Thus, every model that employs B-S variables requires the following set of monotonicity constraints.

$$W_{f\,t-1} - W_{f\,t} \leq 0 \quad for\, all\, t,\, for\, all\, f \tag{3.12}$$

One can see that the standard variables are linearly related to the B-S variables via

$$X_{ft} = W_{ft} - W_{ft-1}\,. \tag{3.13}$$

In [8], Bertsimas and Stock found that the B-S versions of the multi-airport ground holding problem (MAGHP) performed quickly and often offered optimal integer solutions directly from the LP relaxation. According to Bertsimas and Stock, the B-S variables conveniently captured the connecting constraints of the MAGHP and were in many cases facetial in nature. Hoping for similar success with respect to our banking constraints, the transformation (3.13) was applied to models XSS and XGF to obtain models WSS and WGF, respectively.

(WSS)

$$Min \ \sum_{f \in F} \sum_{t=1}^{T} C_f \, (t - a_f)^{\sigma} \, (W_{f\,t} - W_{f\,t-1}) \tag{3.14}$$

$$subject \ to$$

$$W_{f\,T} = 1\,, \ W_{f0} = 0 \quad for\, all\, f \tag{3.15}$$

$$\sum_{f \in F} (W_{f\,t} - W_{f\,t-1}) \leq b_t \quad for\, all\, t \tag{3.16}$$

$$W_{f\,t-1} - W_{f\,t} \leq 0 \quad for\, all\ t,\ for\, all\, f \tag{3.17}$$

$$W_{f\,t} - W_{g \quad t+w_b-1} \leq 0 \quad for,\ all\, t,\ for\, all\ (f,\,g) \in \Phi_b \tag{3.18}$$

$$W_{ft} \in \{0, 1\} \quad for\, all\, f,\ for\, all\, t \tag{3.19}$$

WGF is the same as WSS with the following exceptions: (i) one (ghost flight) binary variable set $\left\{W_t^b : t = 0,\ 1, ..., T\right\}$ is added for each bank b, (ii) the monotone constraint set below is added

$$W_T^b = 1\,, \quad W_0^b = 0 \quad for\, all\, b \tag{3.20}$$

$$W_{t-1}^b - W_t^b \leq 0 \quad for\, all\, t,\ for\, all\, f \tag{3.21}$$

and (iii) constraint set (3.18) is replaced with the following two sets of banking constraints.

$$W_{f\,t} - W_t^b \leq 0 \quad for\,all\,t,\ for\,all\,b,\ \ for\,all\,f \in \Phi_b \tag{3.22}$$

$$W_t^b - W_{f\,t+w_b} \leq 0 \quad for\,all\,t,\ for\,all\,b,\ \ for\,all\,f \in \Phi_b \tag{3.23}$$

Since WSS and WGF are linear transformations of XW and XSS, they will yield the same objective function values (in the LP's) as their assignment variable counterparts. Moreover, since XSS and XGF are equivalent in the LP (see section 4 for proof), the LP optimal function value will be the same for all four models in every problem instance. This fact is confirmed empirically in Tables 1-7, Appendix C.

3.4 Polyhedral Results

The set of integer feasible solutions is the same for each of the models we have presented but the variations in the associated LP relaxations can drastically affect the performance of solution methods based on a branch-and-bound algorithm. Formulations are preferable in which the function value of the LP relaxation is close to the function value of the integer program. In this section, we investigate analytically the strength of the formulations XSS and XGF. We will employ the notation and basic results of polyhedral combinatorics, which can be found Nemhauser and Wolsey [17], and Pulleyblank [20]. We require the following additional notation.

GH = set of integer solutions to constraints (2.2), (2.3) and (2.4), i.e., ground-holding problem

GHB_1 = set of integer solutions to constraints (2.2), (2.3), (2.4) and (3.8), i.e., model XSS

GHB_2 = set of integer solutions to constraints (2.2), (2.3), (2.4), (3.9), (3.10) and (3.11). i.e., model XGF

P^C = convex hull of P , where P is a given set of points in Euclidean Space.

Then GH is the set of feasible solutions to the ground holding problem, GHB_1 is the set of feasible solutions to the double-sum formulation (XSS) and GHB_2 is the set of feasible solutions to the ghost flight formulation (XGF). We will show that, under mild assumptions, each of the banking constraints of the models XSS and XGF represents a facet of its respective polytope. We will show that the each capacity constraint (1.3) represents a facet of both GHB_1^C and GHB_2^C . Finally, we will show that XSS and XGF are equivalent in the strength of their LP relaxations. These results will be based upon the following assumptions.

Assumption 1. $b_T = F$, where F = total number of flights. We assume that the capacity of the last time interval is the same as the number of flights. In would be true in practice to ensure feasible solutions. Our theoretical use of this assumption will be to construct feasible solutions in which an arbitrary number of flights has been assigned to the last time interval without affecting the optimal solution to the problem.

Assumption 2. $\sum_{i=t}^{i=t+w_b} b_i > |\Phi_b|$ for all b and all t,where Φ_b = desired width of bank b. We assume that the capacities of the time intervals are sufficient to allow for the landing of any bank, b, over any block of w_b contiguous time intervals. Combined with assumption 1, this will allow us to generate a feasible solution in which bank b arrives in any chosen block of time intervals and all flights not in bank b arrive in time interval T . The full strength of this assumption is not required but the complexity of the weaker version would obscure the proofs.

Assumption 3. For all t, $b_t \geq 2$. In practice, a time interval would probably represent 10 minutes or more, hence, could accommodate at least two flights. The case in which $b_t < 2$ for some or all of the t might be of theoretical interest.

Assumption 4. We assume that for each flight f, $a_f = 1$, where a_f = scheduled arrival time of flight f. Thus flight f can be assigned to any one of the time intervals, $t = 1, 2, \ldots, T$. This assumption eliminates pathological interactions between the flight arrival times and the bank structure and allows us to index the components of a feasible solution (vector) in the following uniform fashion.

$$X = (X_{11}, X_{12}, ..., X_{1T}, \quad X_{21}, X_{22}, ..., X_{2T}, \quad X_{F1}, X_{F2}, ..., X_{FT}). \qquad (3.24)$$

For notational convenience, let $N = TF$ and $n = TF - F$. We begin by establishing the dimensions of the ambient polytopes.

Lemma 1. *For each constraint C of the form (3.8), there are at least n affinely independent points of GHB_1^C that meet C at equality.*

Proof. See Appendix B. □

Lemma 2. $Dim(GHB_1^C) \geq n$.

Proof. See Appendix B. □

Theorem 3. $Dim(GHB_1^C) = dim(GH) = n$.

Proof. We have already shown that $dim(GHB_1^C) \geq n$. Note that $GHB_1^C \subseteq GH$ and that the constraint set (2.2) contains F linearly independent equations. Therefore, $n \leq dim(GHB_1^C) \leq dim(GH) - F = n$, and the result follows. □

When an instance of GHB is formulated by XGF rather than XSS, one ghost flight is added to the problem for each bank. This increases the number of flights from F to $F+B$, where B is the number of banks. The feasible solutions to XSS are N-dimensional while the feasible solutions to XGF are N^*-dimensional where, $N = TF$ and $N^* = T(F+B)$. Let us assume that, in the formulation of XSS, we have added one ghost flight for each bank. Since the variables corresponding to the ghost flights do not need to appear in the objective function or any of the constraints, the optimal solution will remain unchanged. Now the solution vectors for XSS and XGF are both N^*-dimensional and we can consider the feasibility of a single solution to either XSS or XGF. This simplifies notation and allows us to make use of previous results. In particular, we can restate the conclusion of Theorem 3 to be that $Dim(GHB_1^C) = n^*$, where

$$n* = T(F+B) - (F+B) = n + (TB - B).$$

Theorem 4. *$Dim(GHB_2^C) = n^*$, where $n^* = n + (TB - B)$.*

Proof. Let Ω be the set of all solutions, X and Y, constructed in the proof of Lemma 1. Each solution (vector) in Ω was constructed so that flight 1 lands before all other flights in bank b. Under the assumption that flight 1 is the ghost flight of bank b, each solution in Ω becomes feasible to GHB_2^C . As in Lemma 1, the vectors in Ω can be linearly combined to yield a set, Ω^*, of n linearly (affinely) independent solutions to GHB_2^C . As in the proof of Lemma 2, one more linearly independent vector, U, may be added to Ω^* to bring the total number to (n^*+1). U is formed by setting $U = (Y - X)$, where Y and X are the integer solutions to GHB_2^C , described below.

Let $k = (w_b + t - 1)$.

In block 1: $Y_{1,k} = X_{1,k} = 1$, all other components are zero

In block 2: $Y_{2,k} = 1$, $X_{2,k+1} = 1$, all other components are zero

In block m (m not equal to 1, 2): $Y_{m,n} = X_{m,n}$ for all n. Set these binary components in

any feasible manner.

This shows that $dim(GHB_2^C) \geq n^*$. From Theorem 3, we know that $dim(GHB_1^C) = n^*$ and since $GHB_2^C \subseteq GHB_1^C$, we conclude that $dim(GHB_2^C) = n^*$. $\square$

The following lemma is used to establish that the banking constraints from model XSS induce facets.

Lemma 5. *For every constraint C of the form (3.8), there is an integer point, $X \in GH$, that satisfies every constraint of the form (3.8) except C.*

Proof. Let constraint C be given. This fixes a bank b, a time interval t, and flights $f, g \in \Phi_b$. For notational ease, let us drop the subscripts f and g from the assignment variables $X_{f\,t}$ and $X_{g\,t}$ so we can refer to the variables as X_t and Y_t , respectively. Also, we will assume that both flights are scheduled to arrive in the first time interval so that X_t and Y_t are defined for all t. Then the constraint C is given by

$$\sum_{s=1}^{t} X_s + \sum_{s=t+w_b}^{T} Y_s \leq 1. \tag{3.25}$$

Let S_1 be any solution that assigns $X_t = 1$ and $Y_{t+w_b} = 1$. Since $1+1 > 1$, S_1 violates constraint (3.25) . We will show that S_1 satisfies every other constraint of the form (3.8). Only certain of these constraints apply to the flights f and g and they come in two forms:

$$\sum_{i=1}^{\tau} X_i + \sum_{i=\tau+w_b}^{T} Y_i \leq 1 \tag{3.26}$$

or

$$\sum_{i=1}^{\tau} Y_i + \sum_{i=\tau+w_b}^{T} X_i \leq 1. \tag{3.27}$$

Since each summation in (3.26) and (3.27) is bounded between one and zero, it will suffice to show that exactly one of the two summations is zero. The four cases appear in Appendix B. $\square$

Theorem 6. *Every banking constraint of the form (3.8) represents a facet of GHB_1^C and no two such constraints represent the same facet.*

Proof. Fix a banking constraint, C, and let F be the face represented by C. Lemma 1 shows that there are n^* linearly independent (affinely independent) integer vectors of GHB_1^C that meet C at equality. Thus, $dim(F) \geq n^* - 1$. We know that $dim(GHB_1^C) = n^*$. Let H be the hyperplane represented by C. Since H has dimension greater than n^*, we must consider the possibility that $dim(F) = n^*$. Let GHB_1^{C*} be the polytope that results when constraint C is relaxed from GHB_1^C . By Theorem 3, and the fact that $GHB_1^C \subseteq GHB_1^{C*} \subseteq GH$ we conclude that $dim(GHB_1^{C*}) = n^*$. Now $dim(F) = n^*$ only if all of GHB_1^{C*} lies on H. But 5 shows that a (unique) point of GHB_1^{C*} is eliminated by this hyperplane. Thus, $dim(F) < n^*$. In all, $dim(F) \leq n^* - 1$ and $dim(F) \geq n^* - 1$, so $dim(F) = (n^* - 1)$. F is a facet of GHB_1^C , by definition. It follows from the uniqueness of the point in Lemma 5 that no two such constraints represent the same facet. $\square$

Theorem 7. *Every banking constraint of the form (3.10) represents a facet of GHB_2^C and no two such constraints represent the same facet.*

Proof. See Appendix B. □

Theorem 8. *Every banking constraint of the form (3.11) represents of a facet of GHB_2^C and no two such constraints represent the same facet.*

Proof. Note that every facet of GHB_1^C is also a facet of GHB_2^C . (Recall that we have assumed the existence of ghost flights in the model XSS, so this statement is well defined.) Every ghost-flight constraint of the form (3.11) is a double-sum constraint of the form (3.8). We have already shown that every constraint of the form (3.8) is a facet of GHB_1^C and that the representation is unique. □

Let F_t be the face of GHB_1^C (or GHB_2^C) represented by the capacity constraint corresponding to t. The conditions that are both necessary and sufficient for F_t to be a facet are extremely complex and peculiar to the problem instance. As we will see in the next theorem, a condition sufficient for F_t to be a facet is that there should be at least one solution feasible to all constraints except the capacity constraint. Since GHB is usually being solved under reduced capacity, it would not be hard to construct such a solution. For instance, if flights $f_1, f_2, \ldots, f_{10}$ are scheduled to arrive in time interval t, and if the capacity of time interval t has been cut to, say, $b_t = 7$ flights, then one could assign $f_1, f_2, \ldots, f_7$ to time interval t and all other flights to time interval T. This type of construction would fail for an early time interval for which there are not enough flights to be assigned to it or when there is a bad interaction between bank flights and non-bank flights. For instance, suppose that the only way to fill the capacity of time interval t is to assign a particular flight, f, to time interval t. Then for every (constructed) feasible solution, X, we have the implied equation, $X_{f\,t} = 1$. Since the variables over block f must sum to one, $X_{f\,j} = 0$, for each $j \neq t$. This means that $dim(F_t) < (n^* - 1)$, and F_t cannot be a facet of GHB_1^C (nor of GHB_2^C). But we consider this last scenario to be pathological. The hypothesis of the following theorem would most likely be true in practice.

Theorem 9. *Let F_t be the face of GHB_1^C (or GHB_2^C) represented by the capacity constraint corresponding to time interval t. Then for each $t \neq T$, F_t is a facet of GHB_1^C (or GHB_2^C), provided that there is a set of $b_t + 1$ non-bank flights that can be assigned to t.*

Proof. For each $t \neq T$, one can construct a set of Ω of n^* linearly independent vectors such that each vector $U \in \Omega$ is a linear combination of vectors from F_t (see Appendix B for details of the construction). Therefore, F_t must contain n^* linearly independent vectors. Since linearly independent vectors are affinely independent, it follows that $dim(F_t) \geq (n*-1)$. Recall that $dim(GHB_1^C) = n^* = dim(GHB_2^C)$. Since $F_t \subseteq GHB_1^C$ (and GHB_2^C), we have that $dim(F_t) \leq n*$. Under the assumption that at least $b_t + 1$ flights can be assigned to t, there is at least one feasible solution that does not meet the capacity constraint at equality, hence, does not lie on F_t. Therefore, F_t is a proper subset of GHB_1^C (and GHB_2^C) and we can rule out the possibility that $dim(F_t) = n*$. It follows that $dim(F_t) = (n^* - 1)$ and F_t is a facet by definition. $\square$

By using a polyhedral projection (see [6] and [20] for background), we will show that XSS and XGF are equivalent in strength. Let P_1 be a polyhedron defined over variable set x and let P_2 be a polyhedron defined over variable set (x, z) . Then P_1 is the projection of P_2 onto x if

$$P_1 = \{x : \textit{there exists a } z \textit{ with } (x,z) \in P_2\}.$$

Theorem 10. *Let P_1 be the set of feasible solutions to the LP relaxation of XSS and let P_2 be the set of feasible solutions to the LP relaxation of XGF. Then P_1 is the projection of P_2 onto the variable x.*

Proof. It will suffice to show that (i) whenever $(x,z) \in P_2$, $x \in P_1$

and (ii) whenever $x \in P_1$, there is a z such that $(x,z) \in P_2$.

Proof of (i): Let $y = (x,z) \in P_2$. Fix time interval t and flights f and g in bank b. Because y satisfies every constraint of the form (3.10), we have that

$$\sum_{s=t+1}^{T} Z_s^b - \sum_{s=t+1}^{T} X_{fs} \leq 0. \tag{3.28}$$

The equalities below follow from (2.2) and (3.9), respectively.

$$\sum_{s=t+1}^{T} X_{fs} = 1 - \sum_{s=1}^{t} X_{fs} \tag{3.29}$$

$$\sum_{s=t+1}^{T} Z_s^b = 1 - \sum_{s=1}^{t} Z_s^b \tag{3.30}$$

By substituting (3.29) and (3.30) into (3.28), we obtain

$$\sum_{s=1}^{t} X_{fs} \leq \sum_{s=1}^{t} Z_s^b. \tag{3.31}$$

For an arbitrary flight g in bank b, we add the same expression to each side of (3.31), as below.

$$\sum_{s=1}^{t} X_{fs} + \sum_{ss=t+w_b}^{T} X_{gss} \leq \sum_{s=1}^{t} Z_s^b + \sum_{ss=t+w_b}^{T} X_{gss} \tag{3.32}$$

Since y satisfies every constraint of the form (3.11), the right-hand side of (3.32) and hence the left-hand side of (3.32) is less than or equal to one. We have shown that, for an arbitrary time interval and pair of bank flights, the corresponding constraint of the form (3.10) is satisfied by x. The fact that x satisfies (2.2), (2.3) and (3.32) is trivial. Therefore, $x \in P_1$.

Proof of (ii): Let $x \in P_1$. For each bank b and each time interval t, we define $B_t = MAX_{f \in \Phi_b} \sum_{i=1}^{t} X_{fi}$. For each bank b, we recursively define

$$Z_t^b = \begin{cases} B_t, if\, t = 1 \\ B_t - \sum_{i=1}^{t-1} Z_i^b, \, otherwise \end{cases} \tag{3.33}$$

Let z be the vector whose components are comprised of the variables defined in (3.33). We will show that (x, z) is in P_2 . By definition of Z_t^b , we have that $\sum_{i=1}^{t} Z_i^b = B_t$. Since $0 \leq B_t \leq 1$ for each t, we have that $0 \leq \sum_{i=1}^{t} Z_i^b \leq 1$ for each t. Now whenever $t < \tau$, $B_t \leq B\tau$, so $\sum_{i=1}^{t} Z_i^b$ is non-decreasing, as t increases. Thus, for each t and b, Z_t^b is nonnegative and every constraint of the form $Z_t^b \geq 0$ is satisfied. The feasibility of x to XSS implies that $\sum_{s=1}^{T} X_{fs} = 1$ for every bank flight f and, in particular, $B_T = 1$. Since $\sum_{i=1}^{T} Z_i^b = B_T$, every constraint of the form (3.9) is satisfied for every bank b. These same constraints imply that for every $t > 1$ and every bank b,

$$1 - \sum_{s=t}^{T} Z_s^b = \sum_{s=1}^{t-1} Z_s^b. \tag{3.34}$$

Note that by definition z and $B_t - 1$, $\sum_{s=1}^{t-1} X_{fs} = \sum_{s=1}^{t-1} Z_s^b = B_{t-1}$ for some flight f in bank b with the property that $\sum_{s=1}^{t-1} X_{fs} \geq \sum_{s=1}^{t-1} X_{gs}$ for every g in bank b. Thus, for every g,

$$\sum_{s=1}^{t-1} X_{gs} \leq 1 - \sum_{s=t}^{T} Z_s^b. \tag{3.35}$$

By substituting $\sum_{s=1}^{t-1} X_{gs} = 1 - \sum_{s=t}^{T} X_{gs}$ into (3.35), we obtain the following constraint for every bank flight, g, and every time interval, $t > 1$.

$$\sum_{s=t}^{T} Z_s^b - \sum_{s=t}^{T} X_{gs} \leq 0 \tag{3.36}$$

In the event that $t = 1$, each of the summations in (3.36) is equal to one and the validity of the inequality is trivial. We have shown that (x, z) satisfies every constraint of the form (3.10).

Lastly, to show that (x, z) satisfies every constraint of the form (3.11), fix t and bank b. Let f be the flight corresponding to the maximum sum in the definition of B_t. For every $g \in \Phi_b$, there is a constraint of the following form (3.10) that is satisfied by x. That is,

$$\sum_{i=1}^{t} X_{fi} + \sum_{i=t+w_b}^{T} X_{gi} \leq 1. \tag{3.37}$$

By substituting $\sum_{i=1}^{t} Z_i^b = B_t = \sum_{i=1}^{t} X_{fi}$ in for the left-hand sum in (3.37), we see that (x, z) satisfies constraint (3.11) for an arbitrary t and flight f in an arbitrary bank b. Thus, (x, z) satisfies every constraint of the form (3.11) and $(x, z) \in P_2$, as desired. $\square$

Corollary 11. *The LP relaxations to XSS and XGF have the same optimal objective function values.*

Proof. Note that none of the auxiliary variables (Z_{bt}) appear in the objective function for XGF and that the objective functions for XSS and XGF are the same. The result follows from the preceding theorem. $\square$

3.5 Computational Results

3.5.1 The Data

The performances of the various formulations of GHB (the ground holding problem with banking constraints) were tested on five data sets. Each data set was comprised of a set of flights, a collection of banks (subsets of the set of flights), the scheduled arrival times of the flights, and the capacities of the flights (i.e., the number of passengers that could be carried). The capacities were used to compute the weight of the flight in the objective function.

Data Sets 1 - 4 were constructed with a fictitious airport in mind with an arrival capacity of about one flight per minute. The total number of flights in each data set varied from 25 to 129 and the time horizon varied from 30 minutes to just over two hours. The arrival capacities were designed to mimic those of a large metropolitan airport but the time horizons represent a relatively small slice of time. The time horizons were kept short to be sure that the problems could be solved in a reasonable amount of time. More realistic time horizons would be on the order of 4-6 hours (as in Data Set 5), implying a total of several hundred flights. Each problem instance was solved with a reduced arrival capacity of one-half the original arrival capacity (i.e., one flight every two minutes).

The number of banks per data set was varied from one to seven, each bank consisting of eight to ten flights. In practice, this would be a small or medium-sized bank. The banks were scheduled to land over one to three time intervals. Since the time horizon was divided up into ten minute intervals, this translates to 10-30 minutes per bank. The bank densities (percentage of total flights that were bank flights) ranged from 8.9% to 45%.

We found that when a given data set (1- 4) is solved without banking constraints, each bank would tend to spread over about four time intervals (at ten minutes per time interval, for a total of forty minutes). So, the bank spans were set at three time intervals (thirty minutes total) in order to keep the banking constraints active.

Data Set 5 was actual flight data taken over an eight-hour period at Chicago's O'Hare Airport on February 12, 1993. By default, GDP's are formulated and run over a four hour period so this data set represents a large instance of GHB. We solved the data set over the full eight hour period (13:00 - 20:59, data 5C) but not all the models were able to solve a problem this size, so we generated smaller data sets of four hours (13:00 - 16:59, data set 5A) and six hours (13:00 - 18:59, data set 5B) in order to test fully the performance of each model on real data.

Each problem instance was solved using CPLEX 3.0 on a SPARC 10 work station both as an LP relaxation and as an integer program (IP). We found little or no improvement in performance by customizing the settings provided in CPLEX, so we stayed with the default settings.

With respect to the LP relaxation, we were looking for

- high optimal function values
- low run times, and low iterations of the algorithm

with respect to the IP, we were looking for

- the ability to solve the IP within a node limit of 20,000
- low run times, low number of iterations and low iterations of the algorithm

The computational results are tabulated in Appendix C, Tables 1-7. For each data and for each flight f, the delay constant C_f , was set to one-tenth the passenger capacity of the aircraft. The time intervals were ten minutes each, so the function value units are roughly passenger-delay minutes (they are exactly passenger-delay minutes when the parameter σ is set at 1.0).

3.5.2 The Findings

The value gap of a formulation is the percent by which the LP relaxation optimal value varies from the IP optimal value. A lower value gap indicates a stronger model. In this respect, XGF proved to be the best of the five models. XSS, WSS and WGF will have the same performance relative to this metric since they have equivalent LP's. XGF yielded the lowest value gap in every data set. The value gap for XGF was never more than 2.32% and fell to zero in three of the data sets (1, 5A and 5B), indicating that the optimal integer solution was obtained directly from the LP relaxation.. We believe that the LP strength of the XGF model is due to the fact that each of its banking constraints represents a facet of the convex hull of the set of integer solutions.

Note that for each data set, XGF (but not necessarily XSS, WSS and WGF) solved the IP to integer optimality in very few nodes of the branch-and-bound algorithm (the most was 24 nodes for data set 4).

The run times for XGF (on the IP) varied from fractions of a second to just over 25 minutes (in data set 5B). GDP's are typically formulated a few hours in advance. The specialist would need time to review an optimal solution to GHB before making a final decision, so, in practice, the solution times that XGF displayed would most likely be acceptable.

The outstanding IP performance of XGF comes partly from its LP strength but also from the fact that we greatly reduced the number of nodes required in the branch-and-bound algorithm by relaxing the integer constraints on the assignment variables, X_{ft}. Recall from section 3 that this branching technique was applied not only to XGF but to the other models that use marker variables (to mark the time window in which a bank lands): XWZ and XMMZ. In tables C.1-C.7 (Appendix C), the formulations XWZ, XMMZ are the same as XW and XMM, respectively, but the IP was solved by branching only on the marker (Z) variables. Of course, the LP performances for XW and XWZ are the same (likewise, for XMM and XMMZ). However, the "Z" versions of these models vastly outperformed their counterparts in IP performance. For instance, the number of nodes that XWZ required to solve Data Set 4 was 16 nodes compared to 20,000 for XW.

This difference is so marked that we consider the establishment of marker variables and subsequent branching to be crucial toward solving in real time medium or large instances of GHB (or any such assignment problem with banking constraints).

In every problem instance, the model XMM ranked last in LP strength (i.e., had the highest value gap), run time (both LP and IP) and number of nodes explored in the branch-and-bound algorithm. XMM solved only the smallest of problems (Data sets 1 and 2) to integer optimality in the allotted thresholds of three hours and 20,000 nodes.

3.5.3 Bertsimas-Stock Performance

Our theoretical work has shown that the B-S models are equivalent in LP strength to their standard assignment variable counterparts. Thus, we knew prior to the experiments that they would be equally successful at obtaining integer optimal solutions directly from LP relaxations. So, in performance, we were looking for LP solution times and branching issues.

In general, the B-S versions required more iterations to solve as an LP relaxation - often a full order of magnitude more than their standard counterparts. For instance, WGF required 1683 iterations to solve Data Set 3 (see Table 3, Appendix C) while XGF took only 657. The run times were not so widely different but the standard assignment variable models still outperformed the B-S versions.

For all but the smallest of data sets (i.e., more than 25 flights) the B-S models were outperformed by the standard assignment variable models. One possible reason for the poor performance of the B-S models relates to the replacement of non-negativity constraints with monotonicity constraints (essentially, there is an additional constraint for every variable). This would cause the simplex algorithm to spend significantly more time finding inverses of matrices, thus driving up the LP run times.

We conjectured that the B-S performance would become comparable to the standard versions if the problem had fewer variables. One way to cut down on the number of variables is to limit the amount of delay that could be assigned to any given flight. For instance, if a flight f were scheduled to arrive in the first time interval and there were a total of 25 time intervals, then with a 10 time period limit on the tardiness of each flight, one would need variables W_{ft} for $t = 1, 2, \ldots, 10$ rather than for $t = 1, 2, .., 25$. This type of limitation would

Data Set	Model	Time Int's	Up Bnd	Cap	Iterations	Time (sec)
4	XGF	24	none	7	923	5.63
4	XGF	24	6	7	671	2.17
			Improvement:		27.30%	61.46%
4	WGF	24	none	7	3919	42.00
4	WGF	24	6	7	3067	16.32
			Improvement:		21.74%	61.11%
5A	XGF	30	none	10	3875	80.25
5A	XGF	30	5	10	1450	9.20
			Improvement:		62.58%	88.88%
5A	WGF	30	none	10	12,708	292.30
5A	WGF	30	5	10	7061	83.75
			Improvement:		44.44%	71.35%

Table 3.1: Does LP performance improve with a bound on flight delay?

be done in practice anyway since a flight is effectively canceled if it is severely delayed.

In order to test this hypothesis, we solved the LP relaxation of model WGF on Data sets 4 and 5A, before and after upper bounds of 5 time units and 6 time units, respectively. The results are in Table 3.1.

The runtime of WGF dropped by about 61-72% while the number of iterations dropped by about 22-44% (see Table 8). However, we found comparable savings in run time and iterations (see Table 8) for XGF. The imposed bound did not close the performance gap between the two models.

A very significant property of the B-S models is that very simple constraints tend to represent facets. Recall that every banking constraint of XSS and XGF represented a facet of the convex hull of integer solutions. Since WSS and WGF are linear transformations of XSS and XGF, respectively, the banking constraints of WSS and WGF also represent facets for their respective polytopes. Note that these constraints involve only two variables.

3.5.4 Some Highlights of the Experiments

For Data Set 5, XGF took just over 25 minutes to solve the six-hour time period (13:00 - 18:59, see Data Set 5B) whereas it took XGF only 20 minutes to solve the eight-hour period (13:00 - 20:59, see Data Set 5C). One would think that it would take more time to solve an extension of a problem. We conjecture that the six-hour problem is equally difficult to solve because most of the bank flights are grouped in the first six hours of the eight-hour time period. We further conjecture that the node selection in the branch-and-bound algorithm may have been less fortunate in the six-hour case.

XTC turned out a surprisingly good performance on Data Set 5. Although its LP strength is less than that of XGF (or XSS), it solved Data Sets 5A and 5B in much less CPU time than XGF - sometimes an order of magnitude less. XTC required 1368 nodes of the branch-and-bound algorithm to solve Data Set 5C compared to only 3 for XGF and yet the solution times were comparable (around 20 minutes). This is because XTC was able to solve each iteration of the LP in much less time than XGF. This demonstrates that the strongest model (in LP strength) does not always solve an integer program the fastest.

As one would expect, the length of time required to solve the LP and the IP grows with the time horizon and number of flights. All of the models were able to solve the small Data Sets (1 and 2) in less than a few seconds while on the larger data sets (5A, 5B, 5C) several of the models could not solve the problem in the (arbitrary) three-hour time limit. The relationship between size and run time is not strict, however. Data set 4 is smaller than Data Set 5A (120 flights versus 280 flights) and yet most models (XW and XMM in particular) had far more trouble solving Data Set 4. This might be because Data Set 4 had four banks whereas Data Set 5A had only two.

Chapter 4

The Stochastic Ground-Holding Problem

In Chapter 1, we saw that a GDP (ground delay program) can be formed for a single airport given the schedule of incoming flights and a deterministic arrival acceptance rate, A_t, for each time period t in the planning horizon. Recall that, in essence, the procedure for assigning new arrival times to incoming flights was to list the flights by increasing ETA (estimated time of arrival), then to assign the first A_1 flights on the list to the first time interval, the next A_2 flights to the second time interval and so on.[1]

The effectiveness of a GDP is totally dependent upon the ability of the specialist who formulates the GDP to predict the AAR's (arrival acceptance rates) for each of the time periods. For example, consider the scenario in which a storm is predicted to hit an airport on the east coast at 12:00 hours and, in response, a GDP has been implemented based on the prediction that the arrival capacity for the first hour will drop from 50 to 30 flights per hour. Then the GDP would assign to each of the flights scheduled to arrive during the first hour a ground delay sufficient to ensure that it will arrive without airborne delay. But if the

[1]The complexities of such a formulation will be addressed in Section 4.1.

storm is delayed by several hours or by-passes the airport altogether, then many of these flights will have incurred unnecessary delay. For instance, a flight that was scheduled to leave the west coast at 8:00 and arrive at 13:00 may have been held at its departure airport for one hour so that it would not arrive until after 14:00. Aside from a slight adjustment in enroute air speed, there is no way for this flight to offset its one-hour delay even though, in hindsight, it could have landed on time. In this event, the GDP has assigned too much ground delay.

There is also the opposite scenario in which the arrival capacity of an airport is overestimated and flights incur airborne delay that could have been absorbed on the ground, had a more aggressive GDP been enforced. In this instance, the GDP has not assigned enough ground delay.

It seems that there will always be some uncertainty in the prediction of the arrival capacity at a given airport. AAR's are dependent upon airport configurations which are, in turn, dependent upon meteorological conditions such as visibility, wind velocity/direction, and precipitation. Thus, arrival acceptance rates are stochastic in nature, rather than deterministic.

There are many approaches to stochastic programming (see [9], [16], or [19] for background). The approach we will adopt here is to assume that the time horizon has been discretized into time periods $t = 1, 2, ..., T$ and that for each t, the arrival capacity is a random variable with a discrete probability distribution. We assume that the number, S, of values that the random variable can take on is the same for each time interval. This is consistent with the manner in which airports operate. The AAR is determined based on a small number of key weather parameter and the runway configuration used.

This generates S profiles or, **scenarios** (sample paths), of arrival capacities together with associated probabilities. Figure 4.1 displays possible multiple AAR scenario forecast for a fictitious airport whose normal AAR is 70 flights per hour. The pessimistic forecast, scenario 1, predicts that the AAR will plummet to 30 flights per hour at the third hour and not recover until the eighth hour. Scenario 2 is similar in pattern to scenario 1 but predicts a less severe, shorter storm period that will begin one hour earlier. Scenario 3, the optimistic scenario, predicts heavy impact only in hours 4, 5 and 6, with hour 6 being the worst at an AAR of 55 flights per minute.

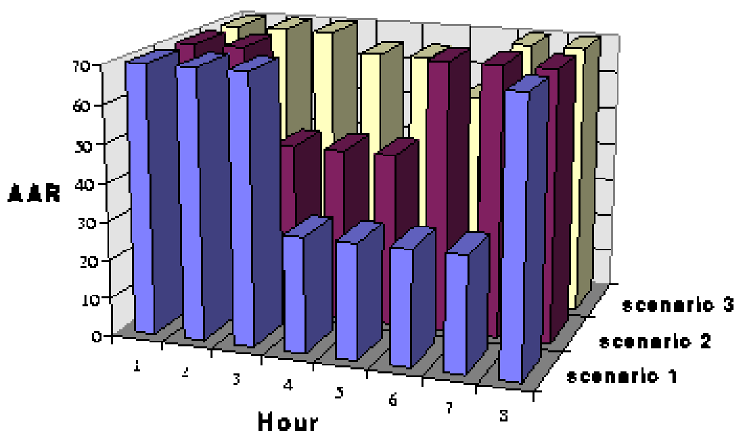

Figure 4.1: Multiple AAR scenarios

Let us consider an intuitive approach to planning AAR's based on the multiple scenarios in Figure 4.1. Suppose that scenario 2 is to occur with high probability, say, $P(2) = .95$. The decision-maker (DM) would do well to set the acceptance rates in close accordance with, if not exactly equal to, this scenario. That is, the DM should (perhaps) accept 60 flights in the first time period, 50 in the second time period, 50 in the third, and so on. However, as the likelihood shifts toward scenario 3, say, $P(1) = .05, P(2) = .55, P(3) = .40$, considerable weight should be given to the optimism of scenario 3 and very little to the pessimism of scenario 1. A GDP based on scenario 2 would probably prove to be overly aggressive and lead to unacceptable levels of unnecessary ground-holding, thus pushing airport demand well below that of capacity. It seems that in the formulation of the GDP, the decision-maker should hedge toward scenario 3 and perhaps disregard scenario 1 altogether.

This type of intuitive reasoning could produce close-to-optimal results for cases in which there are a small number of AAR scenarios with simple patterns. However, there could be a great number of scenarios to consider and it is not uncommon for storm intensity to peak twice or more in an affected region, thus leading to several peaks and valleys in each AAR scenario. This greatly obscures the GDP that minimizes overall delay costs and renders almost useless any simple, intuitive approach for finding it.

The purpose of this chapter is to provide the specialist who formulates a GDP with a modeling tool that will minimize overall expected delay costs while taking into account the stochastic nature of airport arrival capacity. Henceforth, for a given time period, we will be careful to distinguish between the arrival acceptance rate (AAR) and the planned arrival acceptance rate (PAAR). The former is the number of flights that will actually be able to land at the airport while the latter is the number of flights that will attempt to land, based on controlled times of arrival assigned during a GDP. The output of the model will be the PAAR for each time period.

Both the model proposed herein and the use of discrete scenarios are similar to the technique applied by Richetta and Odoni in [22] and Chapter 3 of [23].

Their model yielded a constraint matrix that could be partitioned into network matrices along with coupling constraints. Unable to prove that the formulation would yield an integer solution directly from the LP relaxation, they developed a decomposition algorithm to exploit the special nature of the constraint matrix. It will be shown that the model presented here, however, can be solved by applying standard network code to the dual.

The work presented in this chapter is intended to be used at the end of the CDM (collaborative decision-making) process outlined in the Chapter 2 or at any of its iterative cycles. Since this decision-making process is highly dependent upon human input, it will be modeled as a black box, with final output being an ordering of flights. This ordering of flights is input to the model presented in this chapter, along with the multiple AAR scenarios..

Section 4.1 develops an integer programming model (SGH) of the stochastic ground-holding problem. Section 4.2 presents a simplified version of (SGH_2) in which some of the variables have been eliminated through preprocessing. Section 4.3 addresses the ground-holding and airborne delay costs. The theoretical work in Section 4.4 explores the network structure of the problem and shows that the proposed model allows the integer solution to be obtained directly from its linear programming relaxation. Section 4.5 gives computational results for several large-scale, realistic test cases.

4.1 Model Formulation

Let $t = 1, 2, ..., T$ be a set of discrete, contiguous time intervals during which the AAR of an airport is in jeopardy. Each t could represent, say, a 15-minute time interval. Let $\overline{D}_t$ be the total number of flights scheduled to arrive during time interval t. In practice, not all flights can be given a CTA (controlled time of arrival). International flights and general aviation, for instance, are exempt from ground delay programs in the United States. Let D_t be the total number of (non-exempt) flights that will be included in the GDP. Then the number of exempt flights is given by $E_t = \overline{D}_t - D_t$ with $E_t \geq 0$.

When formulating a GDP, the decision-maker (DM) has the option to exempt flights from the program based on criterion other than those already mentioned. Typically, this criteria is related to the proximity of the origination airport to the destination airport. For instance, when formulating a GDP for an airport on the east coast of the United States, the DM may choose to exempt all long-distance flights, hence, all flights originating on the west coast. So, we assume that the set of flights that are candidates for a given GDP is partitioned into disjoint classes $e = 1, 2, ..., E$, called **tiers**. For each tier, we define a decision variable, y_e, such that $y_e = 1$, if tier e is to be included in the GDP and $y_e = 0$, otherwise. Let d_t^e be the number of flights in tier e. Then we have that

$$D_t = \sum_{e=1}^{E} d_t^e y_e.$$

Let X_t be the PAAR (planned arrival acceptance rate) for time period t, i.e., the number of flights that will be assigned a controlled time of arrival that falls within time period t. This is equivalent to the parameter b_t in the previous chapter and will be the output of our model.

Because of the stochastic nature of weather, X_t flights will not necessarily be able to land during time interval t. For instance, if $X_1 = 10$ and the airport can land only 8 flights at time $t = 1$, then 2 of the 10 flights scheduled to arrive at time $t = 1$ will be held in the air by the controllers at the destination airport. Let us assume that $X_2 = 10$ and that the capacity of the airport is, say, 15 flights for the next period so that the two airborne-held flights are able to land in the next time interval, $t = 2$. Then the airborne delay under this single AAR scenario can be depicted by the flow diagram in Figure 4.2.

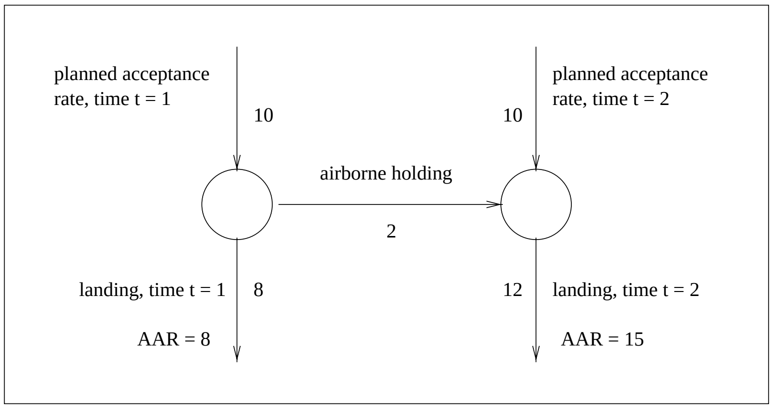

Figure 4.2: Airborne holding of two flights

Assuming that ground holding is cheaper than airborne holding (else there is no need for a GDP), it would have been cheaper to hold two flights on the ground for one time unit and have them arrive at the terminal airspace at time $t = 2$. The single-unit ground delay of these two flights can be represented by adding a horizontal arc at the top of Figure 4.2. The result is shown in Figure 4.3.

Note that Figure 4.3 treats flights on an aggregate level, hence, there is no mention as to which flights will absorb the ground delay. Recall from Chapters 1 and 2 that the flights are sequenced by landing order prior to the formulation of a GDP. Thus, it is easily deduced that the first eight flights are allowed to land at time interval $t = 1$ while the ninth and tenth flights will absorb the ground delay and land in time interval $t = 2$.

Figure 4.3 is easily extended to allow for an arbitrary number of time periods,

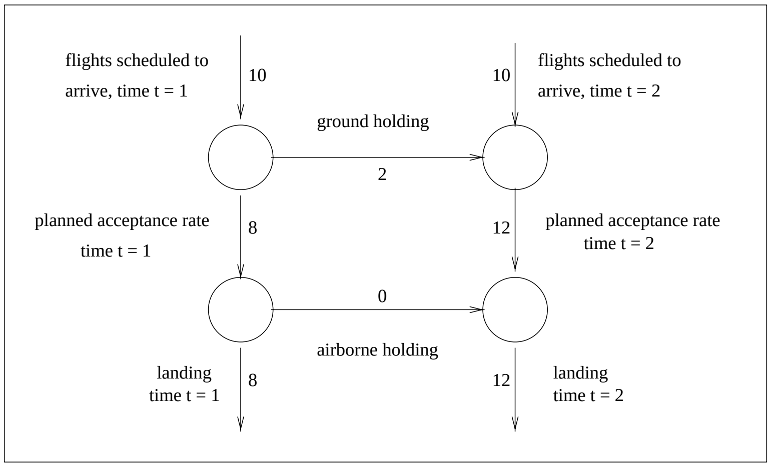
flights scheduled to arrive, time t = 1
10
10
flights scheduled to arrive, time t = 2
ground holding
2
planned acceptance rate time t = 1
8
12
planned acceptance rate time t = 2
0
airborne holding
landing time t = 1
8
12
landing time t = 2

Figure 4.3: Ground holding of two flights

T. Moreover, arcs can been added to allow for the exemption of flights. The result is Figure 4.4, which assumes the following notation.

E_t = number of flights scheduled to arrive at time t that are exempt from ground delay.

b_t^1 = number of flights that can be landed at the airport (the AAR) during time period t.

Z_t = number of (non-exempt) flights wishing to land at time t but not accepted to the terminal airspace at time t (these flights have been ground delayed).

Z_t^1 = number of flights held in the air from time period t to $t+1$ (these flights have been airborne delayed).

The exempted flights flow directly to the bottom nodes, indicating that they will be accepted to the terminal airspace under any conditions. These flights are not exempt, however, from possible airborne holding.

Next, we extend Figure 4.4 to accommodate multiple AAR scenarios. Let $s = 1, 2, ..., S$ be a set of AAR scenarios, each with an associated probability,

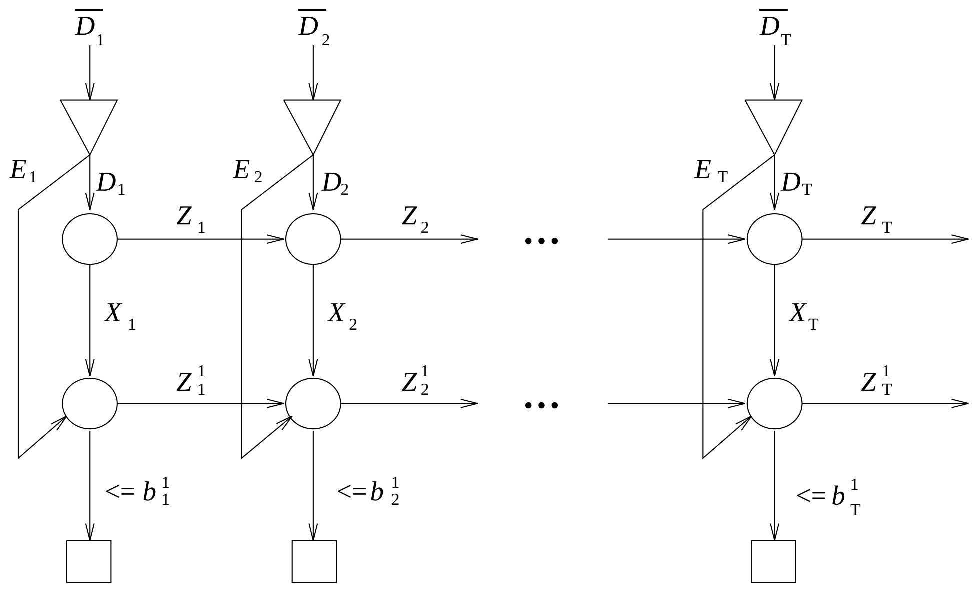

Figure 4.4: Deterministic flow model ($S = 1$)

$p(s)$.The capacity of the airport (AAR) at time t varies with the scenario, s, hence, so does the amount of airborne holding at time t. We generalize the above notation as follows.

Z_t^s = number of flights held in the air from time period t to $t + 1$,under scenario s.

b_t^s = number of flights that can be landed at the airport (the AAR) during time period t,under scenario s.

The lower portion of Figure 4.4 is replicated once for each scenario to arrive at our final diagram in Figure 4.5. Our integer programming model will be based on this diagram.

The hypothetical path of a single flight through the flow diagram is depicted in Figure 4.6. The flight enters the system at time $t = 1$, indicating that it is originally scheduled to arrive at time $t = 1$. Next, the flight absorbs two units of

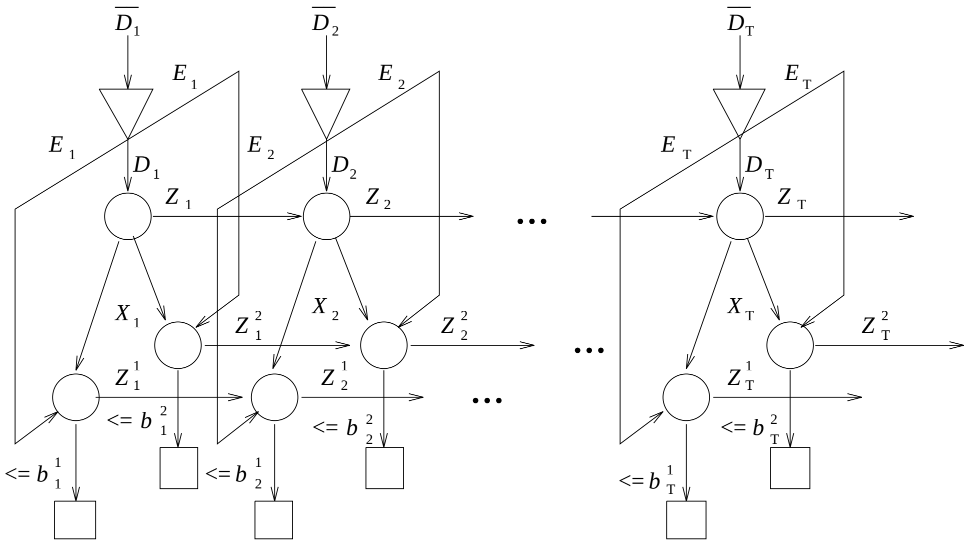

Figure 4.5: Stochastic flow model ($S = 2$)

ground delay, then is allowed to enter the terminal airspace and absorbs two unit of airborne delay. Note that the flow pattern gives the illusion that the flight is being held on the ground just prior to arriving at the terminal space (attempting to land). In reality, it will absorb its ground delay at its origin airport at some earlier time, which is easily computed based on the estimated enroute travel time for the flight.

Let $\tilde{c}$ and $\hat{c}$ be, respectively, the costs of holding a flight for one time unit on the ground and one time unit in the air. Given an outcome (scenario), s, the total delay cost of the GDP is given by

$$\sum_{t=1}^{T} \left(\tilde{c}Z_t + \hat{c}Z_t^s\right),$$

and the expected total delay cost of the GDP is given by

$$\sum_{s=1}^{S} \left[p(s) \cdot \sum_{t=1}^{T} \left(\tilde{c}Z_t + \hat{c}Z_t^s\right) \right]. \tag{4.1}$$

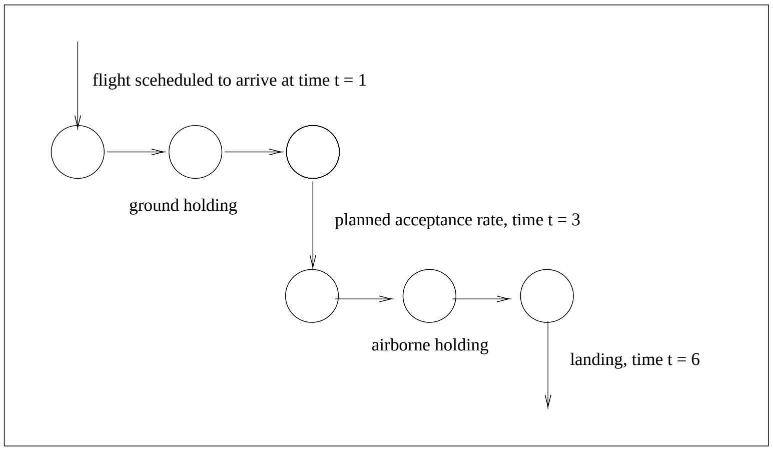
flight sceheduled to arrive at time t = 1
ground holding
planned acceptance rate, time t = 3
airborne holding
landing, time t = 6

Figure 4.6: The path of a single flight

In formulating a GDP, the DM tries to minimize the total expected delay cost subject to the constraints imposed by Figure 4.5. We now develop those constraints.

First, we model the flow through the upper (round) nodes of Figure 4.5. We have the following relations.

$$D_1 = X_1 + Z_1 \tag{4.2}$$

$$Z_{t-1} + D_t = X_t + Z_t \quad for\ all\ t > 1 \tag{4.3}$$

To see that these equations are valid, recall that X_t is the number of flights that will be accepted to the terminal (airport) airspace (but not necessarily land) at time t, that D_t is the number of flights scheduled to land, and that Z_{t-1} is the number of flights denied access to the airport airspace at time $t-1$ (but now wishing to land). Then $Z_{t-1} + D_t$ is the number of flights wishing to land at time t and 4.3 simply says that this number is equal to the number accepted to the airspace at time t (X_t) plus the number rejected from the airspace at time t (Z_t).

We should point out that, unless $S = 1$, the equations of the form 4.2 and 4.3 are not be based on conservation of flow. Note that, for each scenario s, there is one arc flowing out the bottom of any given upper (round) node and that each of these S arcs has flow X_t. This labeling is justified by the fact that once the planned arrival acceptance rate X_t is set, X_t flights are expected to arrive at the destination airport no matter which scenario occurs. (Each of these arcs could have been superscripted with their respective scenarios (X_t^s), in which case we would require coupling constraints of the form $X_t^s = X_t^{s'}$, for $s \neq s'$. The formulation presented here avoids the need for these constraints.) As a consequence, there could be more flow out of one of these nodes than into it.

For convenience, we define u_t^s to be the number of flights that arrive at time t under scenario s (this variable need not appear in the final formulation). This is the flow out the bottom of Figure 4.5 and is limited by the arrival capacity at time t under scenario s, b_t^s. Thus, we have that

$$u_t^s \leq b_t^s \qquad for\ all\ s,\ for\ all\ t. \tag{4.4}$$

By conservation of flow at the lower nodes, variables of the form u_t^s can be defined as follows.

$$u_1^s = X_1 + E_1 - Z_1^s \qquad for\ all\ s \tag{4.5}$$

$$u_t^s = Z_{t-1}^s + X_t + E_t - Z_t^s \qquad for\ all\ s,\ for\ all\ t \geq 2 \tag{4.6}$$

Substituting (4.5) and (4.6) into (4.4), we obtain the following set of constraints.

$$X_1 + E_1 - Z_1^s \leq b_1^s \qquad for\ all\ s \tag{4.7}$$

$$Z_{t-1}^s + X_t + E_t - Z_t^s \le b_t^s \qquad for\ all\ s,\ for\ all\ t \ge 2 \tag{4.8}$$

We define vector $\mathbf{z}$ via

$$\mathbf{z} = \left(Z_1, Z_2, ...Z_T,\ \ Z_1^1, Z_2^1 ..., Z_T^1,\ \ ...\ \ Z_1^S, Z_2^S ..., Z_T^S \right). \tag{4.9}$$

Since the flow across each arc represents a number of flights, this value must be integer and non-negative. In all, we have the following integer program.

(SGH)

$$Minimize\ F(\mathbf{z}) = \sum_{t=1}^{T} \left(\tilde{c} Z_t + \sum_{s=1}^{S} \hat{c} Z_t^s \cdot p(s) \right) \tag{4.10}$$

$$subject\ to$$

$$(4.2), (4.3), (4.7), (4.8)$$

$$Z_{t,} Z_t^s \ge 0\ \ for\ all\ s,\ for\ all\ t$$

$$Z_t,\ Z_t^s\ integer,\ for\ all\ s,\ for\ all\ t$$

This integer programming model yields t constraints of the type (4.3), and $(S \cdot T)$ of type (4.8). There is one constraint of type (4.2) and one of type (4.7). Since the model treats flights on an aggregate level, the total number of constraints is independent of the number of flights and kept quite small: $O(S \cdot T)$. The number of variables is also $O(S \cdot T)$. In an alternative formulation, one could establish, for each t, a set $\{X_t^s : s = 1, 2, ..., S\}$ and a set $\{E_t^s : s = 1, 2, ..., S\}$ and write coupling constraints of the forms $X_t^s = X_t^{s'}$ and $E_t^s = E_t^{s'}$, for every pair $s \neq s'$.

For the case $S = 1$, the model is deterministic and the diagram in Figure 4.5 reduces to the diagram in Figure 4.4. The deterministic case $S = 1$ is easily solved by the following greedy algorithm. For $t = 1, 2, ..., T$, land as many flights as possible (but no more than b_t^1) from the exempt and airborne-delayed categories of flights (reflected in the variables E_t and Z_{t-1}^1), then carry over the remainder into the next time period in the form of airborne holding (via the variable Z_{t+1}^1). This reduces the capacity of the airport to some number, b_t^*. Then, if $b_t^* > 0$, land b_t^* (or as many as possible) of the non-exempt flights (reflected in the variables $\tilde{Z}_{t-1}$ and D_t) via the variables X_t and u_t^1. Any excess of non-exempt flights must be held on the ground and is reflected in the variable Z_t (assuming that airborne holding is more expensive than ground holding).

When $S \geq 2$, the problem is truly stochastic and the greedy algorithm is ill-defined because the phrase, "land as many flights as possible" becomes ambiguous (which of the capacities $b_t^1, b_t^2, ..., b_t^S$ for time t should one use?).

The model resembles a classic production-inventory model in which an inventory can be held in one of two states (see Section 4.5 of [14]). The flights correspond to a product or materials, such as crude oil. A quantity of materials (the flights) is purchased when it enters the diagram at a time period and can then be held in raw form (ground-holding) or finished form (airborne holding). The materials are sold when they exit the diagram at the bottom. But the model presented here is distinguished in three ways. First, it does not seek production levels to be set for each time period; these levels are set in advance by the parameters of the form D_t. Second, and more crucially, the product can be held over (in two different types of inventory) at two different costs, as in a raw material/finished good production model, such as crude oil/refined oil. However, the amount held over in the second form is stochastic, depending on a random variable. This prevents the model from being solved by standard techniques. (See [19] for a treatment of stochastic network routing.) Third, the model cannot be translated into a network model, even though it is depicted by a flow diagram. (The dual of the model, however, can be, as shown in Section 4.4.)

4.2 Pre-processing

In this section we show that for each tier e and each time interval t, the variables y_e and E_t can be eliminated from the formulation of SGH by pre-processing.

Note that whenever a tier e is excluded from the GDP, this shifts the flights of tier e from the parameter D_t to the parameter E_t and one has lost the ability to assign ground delay to these flights. There are many reasons why this should be done in practice, but from an optimization standpoint, there is no advantage to this type of exclusion because it can only decrease the flexibility of the program. Specifically, k flights can be effectively exempted from ground delay at time t by increasing the variable X_t by k units. Proposition 12 is the formalization of this observation.

Proposition 12. *For every feasible solution,* $\mathbf{z}$*, to* SGH*, there is a feasible solution,* $\mathbf{z}^*$*, with the properties that* $F(\mathbf{z}^*) = F(\mathbf{z})$ *and* $y_e = 1$ *for* $e = 1, 2, ..., E$*.*

Proof. Let $\mathbf{z}$ be any feasible solution to SGH with corresponding function value $F(\mathbf{z})$. If $y_e = 1$ for every e, then we set $z = z^*$ and there is nothing to show.

Otherwise, we create a new solution, $\mathbf{z}^*$, as follows. For each index e such that $y_e = 0$ in solution $\mathbf{z}$, we set $y_e = 1$. By definition D_t, this increases the value of D_t by the amount d_t^e and decreases the value of E_t by the same amount. This jeopardizes equality in equation (4.3), but equality can be maintained by adding the amount d_t^e to the value of X_t. This also preserves feasibility in constraints (4.8). Since the values of Z_t^s and Z_t remain unchanged for every s and every t, we have created a new feasible solution, $\mathbf{z}^*$ with the same function value as $\mathbf{z}$. The result follows from the fact that y_e does not appear in the objective function and that the index e was chosen arbitrarily. $\square$

As a consequence of Proposition 12, we can make the simplifying assumption that $y_e = 1$ for $e = 1, 2, ..., E$ and we discard these variables from the formulation of the problem. In practice, this means that the values of the variables y_e are set a priori by the DM (as opposed to being output by the solution) and that the values of E_t and D_t are adjusted accordingly.

An alternate model could be developed in which the exclusion or inclusion of tiers from the formulation of the GDP is not so rigid. The variables y_e could appear in the objective function with associated penalties. Each penalty would reflect the reluctance of the decision-maker to include the associated tier in the program. Such a model would be useful, for instance, whenever a tier is comprised of flights that have been de-iced and should not be delayed. The above proposition would not hold in such a model.

Now we consider the variables of the form E_t. We wish to show that these variables can be pre-processed and subsequently eliminated from the formulation of the problem. In light of constraints (4.8), the effect of increasing E_t is to subtract from the capacity parameter b_t^s for each s. This is the same as saying that, under any circumstances, the exempt flights will be entering the terminal airspace during time interval t thereby reducing arrival capacity. So in the formulation of SGH, the exempt flights can be pre-processed via $b_t^s := b_t^s - E_t$, for each s and each t. But whenever $E_t > b_t^s$, this pre-processing becomes meaningless and, worse yet, constraints (4.8) become infeasible and the solution set to SGH is empty. In practice, the number or exempt flights (this includes international flights and those already airborne at the time of the formulation of the GDP) is usually on the order of ten percent and is not likely to exceed the AAR of the airport, even under severe weather conditions. However, we already granted the DM with the option to exempt flights by setting the values of some of the tier variables at $y_e = 0$. Thus, the number of exempt flights could easily exceed the AAR in some, or perhaps all, of the scenarios and we should not too readily dismiss the possibility that $E_t > b_t^s$ for some t. In this event, the pre-processing that needs to be done is to set $b_t^s := 0$ and let the excess of exempt flights be carried over into the next set of exempt flights at time $t + 1$. Formally, this pre-processing is defined as follows. For $t = 1, 2, ..., T$, we recursively define new values of E_t and b_t^s, denoted by $\overline{E}_t$ and $\overline{b}_t^s$, respectively, as follows.

$$\bar{E}_t = \begin{cases} E_t, \quad if\ t = 1 \\ E_t + max\left(0, \bar{E}_{t-1} - b_{t-1}^s\right), \ if \ \ t > 1 \end{cases}$$

$$\bar{b}_t^s = max\left(0, b_t^s - \bar{E}_t\right)$$

Now we can proceed under the assumption that for each t, $E_t = 0$ or, in other words, $\overline{D}_t = D_t$. Combining this with the prior assumption that $y_e = 1$ for every e, we have the following revision of SGH.

(SGH_2)

$$Minimize\,(4.10)$$

$$subject\ to$$

$$D_1 = X_1 + Z_1 \tag{4.11}$$

$$Z_{t-1} + D_t = X_t + Z_t \quad for\ all\ t \tag{4.12}$$

$$X_1 - Z_1^s \leq b_1^s \qquad for\ all\ s \tag{4.13}$$

$$Z_{t-1}^s + X_t - Z_t^s \leq b_t^s \qquad for\ all\ s,\ for\ all\ t \tag{4.14}$$

$$Z_t, Z_t^s \geq 0 \qquad for\ all\ s,\ for\ all\ t \tag{4.15}$$

$$Z_t,\ Z_t^s\ integer\ for\ all\ s,\ for\ all\ t \tag{4.16}$$

4.3 Holding Costs: The Efficient Frontier

An exact calculation of the cost of ground-holding or airborne holding for any given flight involves complex factors such as crew costs, fuel consumption, and connectivity with other flights. It is be hard to assign a dollar value to each minute of passenger delay and still harder to determine whether this delay is more costly when taken on the ground or in the air. A proper assessment of delay costs should also include the risk for passenger safety that is incurred in airborne delay, and the connectivity between flights: a flight that is unnecessarily delayed on the ground will, in turn, delay any flights that connect with it at its destination airport.

The assessment of air and ground delay costs is extremely complex, situation-specific and, in many cases, highly subjective. Moreover, much of the information required for a careful calculation of holding costs is currently unavailable to the FAA. This makes it exceedingly impractical for the FAA to assign holding costs to individual flights in real time.

One solution to this cost-assessment dilemma is to average delay costs so that both airborne holding costs and ground holding costs are uniform for all flights. The Air Transportation Association [1] estimates that for a typical flight, every minute of ground delay costs \$20.35 while every minute of airborne delay costs \$45.85. Our model is in keeping with, but not limited to, this approach. An alternative to this proposed by Richetta in [21] is to assume that flights can be grouped into a small number or classes, each with one delay cost.

Our model allows for an interpretation of delay costs that largely avoids the economics of airline operations - an interpretation which we now explore.

The primary mission of the FAA, hence, the ATCSCC (air traffic control systems command center), is to ensure passenger safety. This might tempt one to conclude that all GDP′s should be aggressive enough to ensure that, under any reasonable circumstances, no flight will be subjected to airborne holding. However, it is also the task of the FAA to manage resources within the NAS (national air space) in an efficient, even-handed manner. Any such policy would, on average, lead to a gross under-utilization of airport arrival capacity and the services provided by the air traffic controllers. The DM must strike a balance between restricted traffic flow and efficient allocation of resources.

In so doing, the decision maker at the ATCSCC must address airline needs and perceptions. In particular, an airline finds it aggravating when a GDP is aborted in mid-operation. This happens whenever the DM has clear evidence that the original capacity and weather forecasts were overly pessimistic. Of course, the GDP was unnecessary only in hindsight but it is hard to justify ground delay to passengers who see "blue-sky" conditions at their departure airport. For this reason, the airlines are frequently willing to risk small or moderate amounts of airborne delay and tend to favor more liberal ground-holding policies.

The DM may require the freedom to vary the values of the costs $\tilde{c}$ and $\widehat{c}$ with the airport and situation at hand. Because our model is solved so rapidly, the decision maker can retain the option to explore a number of cost pairs, $(\tilde{c}_1, \widehat{c}_1)$, $(\tilde{c}_2, \widehat{c}_2)$, ..., $(\tilde{c}_n, \widehat{c}_n)$. Let us assume that the ground holding costs have been normalized to 1.0 by forming the ratio $\kappa_i = \hat{c}_i/\tilde{c}_i$, for each cost pair.

By plotting on one graph each expected delay cost, $f(\kappa_i)$, as a function of its cost ratio κ_i the decision-maker can establish an **efficient frontier** for the formulation of the GDP (see Figure 4.7). For a given cost ratio, κ_i, any proposed GDP that is sub-optimal will lie to the upper left of the frontier. The effect of our model is to help the decision-maker push the GDP back onto the frontier.

Our alternative perspective, then, on the delay cost ratio (air to ground) is that of a numerical expression of the willingness of the DM and the ATCSCC to trade ground holding for airborne holding - the higher the cost ratio, κ, the more conservative will be the resulting optimal GDP. Any further treatment of values for delay costs would be venturing into the arena of policy-making and well beyond the scope of this technical paper. (For sensitivity analysis of the model with respect to delay cost parameters, see section 4.5.)

Lastly, we point out that the structure of the objective function allows the delay costs, both airborne and ground, to vary with t. This grants the decision maker the flexibility to establish time intervals of varying length. One possible use would be to have time periods of increasing length so that short-term planning is done on a refined level while long-range planning is more coarse.

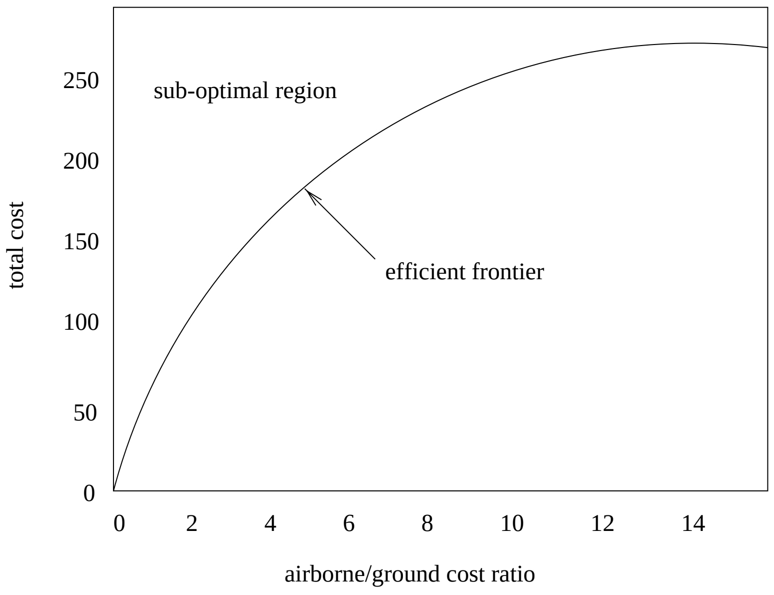

Figure 4.7: The efficient frontier

4.4 Theoretical Results

We say that a $(0, 1, -1)$ matrix is in **network matrix form** if each column contains at most two nonzero entries and whenever a column contains two nonzero entries, they sum to zero. If N is in network matrix form, then one can solve the following LP as a network flow problem.

$$Max\ c^T x \tag{4.17}$$

$$subject\ to$$

$$Nx = b$$

$$x \geq 0$$

Specifically, by adding a redundant equation (the negative sum of the equations) to the system in (4.17), one obtains an equivalent system, $N'x = b'$, in which the matrix N' is a node-arc incidence matrix for an underlying directed graph G over which one is trying to maximize flow. The system $N'x = b'$ is comprised of conservation of flow and arc capacity equations for the nodes in G.

Network flow problems are desirable because they can be solved via known algorithms which are, in many cases, faster than the simplex procedure. (See [2] for background.) But these problems often appear in hidden form. Suppose that a sequence of simplex pivots is performed on the system of equations in (4.17), where a single simplex pivot is defined to be the row operations necessary to obtain $a_{ij} = 1$ in the i^{th} column of the j^{th} row, for some i, j, and to obtain $a_{ik} = 0$ for all $k \neq j$. In the new system, $N''x = b''$, it is highly unlikely that N'' is in network matrix form. So, if one starts with this $N''x = b''$ it might not be obvious that it can be transformed (back) into a network flow problem and even less obvious how to do so.

This prompts the following definition. Let ν be the set of all matrices in network matrix form. We define the set of network matrices, denoted $C(\nu)$, to be the closure of ν under simplex pivoting. That is, an $m \times n$ matrix $M \in C(\nu)$ iff there is an $m \times n$ matrix $N \in \nu$ such that M can be derived from N by a sequence of simplex pivots. For each network matrix, M, the sequence of simplex pivots by which M is obtained from some N can be represented by an invertible matrix, E, such that $M = EN$. In this case, we say that M is a **network matrix with respect to** N **via** E. Note that every network matrix is a $(0, 1, -1)$ matrix. See [17] for background material on network matrices.

If M is a network matrix with respect to N via E, and if either N or E is known, then the LP

$$\begin{gathered} Max\ c^T x \\ subject\ \ to \\ Mx = b \\ x \geq 0 \end{gathered} \tag{4.18}$$

can be transformed into, hence solved as, a network flow problem. In this section, we will be exploring the possibility that either the primal or dual of SGH_2 is a network flow problem in hidden form. We will be considering only those cases in which $S \geq 2$, for if $S = 1$, SGH_2 is deterministic and solved by a greedy algorithm. In fact, it is easy to show that the primal problem is a network flow problem, when $S = 1$.

The primal of SGH_2 is given by

$$\begin{gathered} Min\ c^T x \\ subject\ \ to \\ Ax \geq b \\ x \geq 0 \end{gathered} \tag{4.19}$$

where

$$c^T = \left(Block\ repeats\, T\ times \underbrace{\tilde{c}, \hat{c}_1, \hat{c}_2, ..., \hat{c}_S, 0, \quad ... \quad , \tilde{c}, \hat{c}_1, \hat{c}_2, ..., \hat{c}_S, 0} \right), \tag{4.20}$$

$$\hat{c}_k = p(k) \cdot \hat{c}, \tag{4.21}$$

$$x = \left(Z_1, Z_1^1, Z_1^2, ..., Z_1^S, X_1, \quad Z_2, Z_2^1, Z_2^2, ..., Z_2^S, X_2, \quad ... \quad , Z_T, Z_T^1, Z_T^2, ..., Z_T^S, X_T \right), \tag{4.22}$$

and

$$b^T = \left(D_1, b_1^1, b_1^2, ..., b_1^S, \quad D_2, b_2^1, b_2^2, ..., b_2^S, \quad ... \quad , D_T, b_T^1, b_T^2, ..., b_T^S \right). \tag{4.23}$$

For instance, when $S = 2$ and $T = 2$, the primal is as below.

$$min \left(\tilde{c}Z_1 + \hat{c}_1 Z_1^1 + \hat{c}_2 Z_1^2 + \tilde{c}Z_2 + \hat{c}_1 Z_2^1 + \hat{c}_2 Z_2^2 \right) \tag{4.24}$$

$$subject\ to$$

$$\begin{array}{ccccccccc}
Z_1 & & & X_1 & & & & \geq & D_1 \\
& Z_1^1 & & -X_1 & & & & \geq & -b_1^1 \\
& & Z_1^2 & -X_1 & & & & \geq & -b_1^2 \\
\text{-}Z_1 & & & & Z_2 & & X_2 & \geq & D_2 \\
& \text{-}Z_1^1 & & & & Z_2^1 & -X_2 & \geq & -b_2^1 \\
& & \text{-}Z_1^2 & & & Z_2^2 & -X_2 & \geq & -b_2^2
\end{array}$$

After adding surplus variables to the system in (4.19), we obtain the system of equations $(I, A)x = b$ and we need to determine whether or not (I, A) is a network matrix. There are network recognition algorithms for determining this.

See [10] or [17] for a reference. The core of any such algorithm relies on the following key facts. Suppose that M is an $m \times n$ network matrix with respect to N via E and that and each column of the $m \times m$ identity matrix appears exactly once in M. Since N is a node-arc incidence matrix (minus one row) there is an underlying graph, G, of n edges on $m+1$ nodes. The appearance of an identity column in the j^{th} column of M indicates that the j^{th} edge of G is in a basis B of m edges that form a spanning tree for G. By association through the identity columns, each row of M corresponds to an edge of B. Each non-identity column is the characteristic vector of a path in G, with respect to the edges in B.

Lemma 15 in Appendix B shows that for every $S \geq 2$, the primal matrix (I, A) of SGH_2 fails to be a network matrix. The lemma derives a contradiction from the (assumed) coexistence of the aforementioned paths without directly appealing to the recognition algorithms, though they will reveal the same contradiction.

We turn our attention to the dual of SGH_2, given below.

$$\begin{gathered} Max\ b^T w \\ subject\ \ to \\ A^T w \leq c \\ w \geq 0 \end{gathered} \tag{4.25}$$

A^T is an $m \times n$ matrix where $m = T(S+2)$ and $n = T(S+1)$. We begin with an explicit block description of A^T.

Let D and E be the $(S+2) \times (S+1)$ matrices defined via

$$d_{ij} = \begin{cases} 1,\ if\ (i=j)\ or\ (i = S+2\ and\ j=1) \\ -1,\ if\ i = S+2\ and\ j \neq 1 \\ 0,\ otherwise \end{cases}, \qquad e_{ij} = \begin{cases} -1,\ if\ i=j \\ 0,\ otherwise \end{cases}.$$

Then A^T is the $T \times T$ block matrix such that the $(i,j)^{th}$block is of size $(S+2) \times (S+1)$ and defined by

$$A^{ij} = \begin{cases} D, \ if \ \ i = j \\ E, \ \ if \ \ i+1 = j \\ 0 \ matrix, otherwise \end{cases} .$$

For example, when $S = 2$ and $T = 2$,

$$A^T = \begin{bmatrix} D & E \\ 0 & D \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 \\ 1 & -1 & -1 & 0 & 0 & 0 \\ & & & 1 & 0 & 0 \\ & & & 0 & 1 & 0 \\ & \text{-0-} & & 0 & 0 & 1 \\ & & & 1 & -1 & -1 \end{bmatrix} .$$

We wish to show that the $m \times (m+n)$ matrix $M = \left(I, A^T\right)$ is a network matrix. We will construct an $m \times (m+n)$ matrix N in network matrix form such that N can be transformed into M via simplex pivots. Let Δ, Δ', be $(S+2) \times (S+2)$ matrices defined via

$$\delta_{ij} = \begin{cases} 1, \ if \ (i = 1 \ \ and \ \ j = 1) \ or \ (i = 2 \ and \ j \geq 2) \\ -1, \ \ if \ \ i - 1 = j \\ 0, \ \ otherwise \end{cases} ,$$

$$\delta'_{ij} = \begin{cases} -1, \ if \ i = 2 \ and \ \ j = S+2 \\ 0, \ \ otherwise \end{cases} ,$$

and let ε be the $(S+1)\times(S+1)$ matrix defined by

$$\epsilon_{ij} = \begin{cases} 1, \; if \;\; i = 1 \;\; and \;\; j = 1 \\ -1, \;\; if \;\; i \geq 3 \;\; and \;\; i-1 = j \\ 0, \;\; otherwise \end{cases} .$$

We define N to be the $T \times 2T$ block matrix below.

$$N = \begin{bmatrix} \Delta & \Delta' & & & & \varepsilon & -\varepsilon & & & \\ & \Delta & \Delta' & & & & \varepsilon & -\varepsilon & & \\ & & \dots & \dots & & & & \dots & \dots & \\ & & & \Delta & \Delta' & & & & \varepsilon & -\varepsilon \\ & & & & \Delta & & & & & \varepsilon \end{bmatrix} \tag{4.26}$$

For example, when $S = 2$ and $T = 2$,

$$\Delta = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 1 & 1 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix} \quad \Delta' = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \varepsilon = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

and

$$N = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 \\ -1 & 1 & 1 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 \\ & & & & 1 & 0 & 0 & 0 & & & & 1 & 0 & 0 \\ & \text{-0-} & & & -1 & 1 & 1 & 1 & & \text{-0-} & & 0 & 0 & 0 \\ & & & & 0 & -1 & 0 & 0 & & & & 0 & -1 & 0 \\ & & & & 0 & 0 & -1 & 0 & & & & 0 & 0 & -1 \end{bmatrix} \tag{4.27}$$

In this instance, N is clearly in network matrix form. Lemma 16 in Appendix B shows that, for any $S \geq 1, T \geq 1$, N is in network matrix form. It is easily verified that when simplex pivots are performed successively on elements $n_{1,1}$, $n_{2,2}$, ..., $n_{m,m}$, then N is transformed into $M = (I, A)$. For example, the above instance of N is transformed into the matrix below.

$$M = (I, A) = \begin{bmatrix} 1 & 0 & 0 & 0 & & & & & 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & & \text{-0-} & & & 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & & & & & 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 & & & & & 1 & -1 & -1 & 0 & 0 & 0 \\ & & & & 1 & 0 & 0 & 0 & & & & 1 & 0 & 0 \\ & \text{-0-} & & & 0 & 1 & 0 & 0 & & \text{-0-} & & 0 & 1 & 0 \\ & & & & 0 & 0 & 1 & 0 & & & & 0 & 0 & 1 \\ & & & & 0 & 0 & 0 & 1 & & & & 1 & -1 & -1 \end{bmatrix} \tag{4.28}$$

Let E be the matrix (established by simplex pivots) that transforms N into M. The inverse of E is the matrix that reverts M to network matrix form. Since $E^{-1}M = N$ and $M = (I, A)$, we have that $N = (E^{-1}, E^{-1}A)$. So, E^{-1} must be comprised of the first $m = T(S+2)$ columns of N and E^{-1} must be in network matrix form. Moreover, since we have an explicit description of N, we have an explicit description of the transformation E^{-1}.

In all, we have established the following theorem.

Theorem 13. *Let A^T be the symmetric dual matrix of the LP relaxation of SGH_2. Then $M = (I, A^T)$ is a network matrix.*

Corollary 14. *The primal matrix of the LP relaxation of SGH_2 is integral, whenever the problem is feasible.*

Proof. The dual matrix is a network matrix, hence, totally unimodular (TU). Since every TU matrix is integral, both the primal and dual of $SGH_2$are integral under the assumption that the problem is feasible. □

As a result of the above corollary, the integer solution to SGH_2 can be obtained directly from its LP relaxation.

Some insight into SGH_2 can be gained by examining the dual and the induced network. We resume the example in which $S = 2$ and $T = 2$. The symmetric dual is given below.

$$Max\ \ D_1w_1 - b_1^1w_1^1 - b_1^2w_1^2\ + D_2w_2 - b_2^1w_2^1 - b_2^2w_2^2 \tag{4.29}$$

subject to

$$
\begin{array}{cccccccl}
w_1 & & & -w_2 & & & \leq & \widetilde{c} \\
& w_1^1 & & & -w_2^1 & & \leq & \widehat{c}_1 \\
& & w_1^2 & & & -w_2^2 & \leq & \widehat{c}_2 \\
w_1 & -w_1^1 & -w_1^2 & & & & \leq & 0 \\
& & & w_2 & & & \leq & \widetilde{c} \\
& & & & w_2^1 & & \leq & \widehat{c}_1 \\
& & & & & w_2^2 & \leq & \widehat{c}_2 \\
& & & w_2 & -w_2^1 & -w_2^2 & \leq & 0
\end{array}
$$

Let $(I, A)(w) = c$ be the system of equations obtained after adding surplus variables to (4.29). Let E^{-1} be the transformation matrix comprised of the first eight columns of the matrix in (4.27). After applying the transformation E^{-1} to $(I, A)(w) = c$, we obtain the following network flow problem.

$$Max \;\; D_1 w_1 - b_1^1 w_1^1 - b_1^2 w_1^2 \; + D_2 w_2 - b_2^1 w_2^1 - b_2^2 w_2^2 \tag{4.30}$$

$$subject \;\; to$$

$$
\begin{aligned}
s_1 + w_1 - w_2 &= \widetilde{c} \\
-s_1 + s_2 + s_3 + s_4 - s_8 &= \tilde{c} - \hat{c} \\
-s_2 - w_1^1 + w_2^1 &= -\widehat{c}_1 \\
-s_3 - w_1^2 + w_2^2 &= -\widehat{c}_2 \\
s_5 + w_2 &= \widetilde{c} \\
-s_5 + s_6 + s_7 + s_8 &= \tilde{c} - \hat{c} \\
-s_6 - w_2^1 &= -\widehat{c}_1 \\
-s_7 - w_2^2 &= -\widehat{c}_2
\end{aligned}
$$

Each of the equations in (4.30) is a conservation of flow equation or arc capacity equation for one of the nodes in the underlying graph G, depicted in Figure 4.8. The equations of the system $(I, A)(w) = c$ correspond to the following primal variables, respectively.

$$Z_1, Z_1^1, Z_1^2, X_1, Z_2, Z_2^1, Z_2^2, X_2 \tag{4.31}$$

For the sake of consistency, we have maintained this labeling system for the equations in (4.30), even though each equation is actually a linear combination of the equations in $(I, A)(w) = c$ (i.e., the dual network variables of the network flow problem are not logically equivalent to the original problem variables). Hence, each node in G is labeled with the appropriate primal variable except for the left-most node, which is superfluous to the problem. Flow is conserved at this node by the redundant equation formed by the negative sum of the equations in (4.30).

We may assume that $\tilde{c}$, $\hat{c}$, and $\hat{c} - \tilde{c} > 0$, or else SGH_2 can be solved by a greedy algorithm and there is no need to appeal to the dual or the network. Under this assumption, each of the equations in (**??**) contains a nonzero constant, indicating that the corresponding node is a source node or sink node, depending if the constant is positive or negative. Nodes Z_1, Z_2, Z_1^1, Z_2^1 are sources with values of $\tilde{c}, \tilde{c}, \hat{c}-\tilde{c}, \hat{c}-\tilde{c}$, respectively, while the nodes X_1, X_2, Z_1^2, Z_2^2 are sinks with values of $\hat{c}_1, \hat{c}_1, \hat{c}_2, \hat{c}_2$, respectively. For purposes of illustration, we have connected the source nodes in Figure 4.8 to a single source and connected the sink nodes to a single sink. The connecting arcs are capacitated by the respective values just listed. All other arcs in the diagram are uncapacitated.

Recall that the for $t = 1, 2$ the input parameter D_t is the number of flights that enter the system (are scheduled to arrive) in time periods t and that the capacity parameters b_t^s, is the maximum numbers of flights that can exit the system (arrive) at time t under scenario s. The (positive) costs of the flows along the upper two arcs, w_1 and w_2, are set by the input parameters D_1 and D_2 while the costs of the flows along the lower four arcs w_1^1, w_1^2, w_2^1 and w_2^2, are set by the negatives of the capacity parameters, b_1^1, b_1^2, b_2^1 and b_2^2. There is no cost to flow along the arcs corresponding to the surplus variables, $s_1, s_2, .., s_8$.

In this network, flow must be set so as to maximize revenue, subject to conservation of flow at each node and the capacities on the arcs flowing in and out of the sources and sinks. Since arc capacity is given by the cost parameters $(\tilde{c}, \hat{c})$ of the primal, revenue can be measured in dollars but there doesn't appear to be any meaningful interpretation of the dual with respect to the primal.

For certain special cases, it is easy to see that the network flow problem depicted in Figure 4.8 gives the same optimal function value as the primal. For instance, if the airport capacity b_t^s in the primal is zero for each s and each t, then the optimal solution to the network is to allow maximal flow along the upper arcs, that is, $w_2 = \tilde{c}$ and $w_1 = 2\tilde{c}$, at a cost of $\tilde{c}D_1 + 2\tilde{c}D_2$. In the primal, since there is no arrival capacity at the airport at any time period under any scenario, the optimal solution is to ground hold each flight for each time period, that is, $Z_1 = D_1$ and $Z_2 = D_1 + D_2$ at the same cost of $\tilde{c}Z_1 + \tilde{c}Z_2 = \tilde{c}D_1 + \tilde{c}(D_1 + D_2) = \tilde{c}D_1 + 2\tilde{c}D_2$.

It would be hard to imagine a situation in which the optimal solution would allow flow along arcs s_4 and s_8, for this could only incur negative cost. This intuition is confirmed by complementary slackness. Under any ordinary circumstances, the optimal solution to the primal will probably allow the landing of a positive number of fights in time intervals 1 and 2. In this event, the values of X_1

and X_2 are positive. By complementary slackness, we know that $X_1 \cdot s_4 = 0$ and $X_2 \cdot s_8 = 0$, so both s_4 and s_8 must be zero in any optimal solution. Moreover, if there is a positive amount of ground holding in time period 1 (or 2) of the optimal, then there will be no flow along arc s_1 (or s_5) in the optimal solution to the dual.

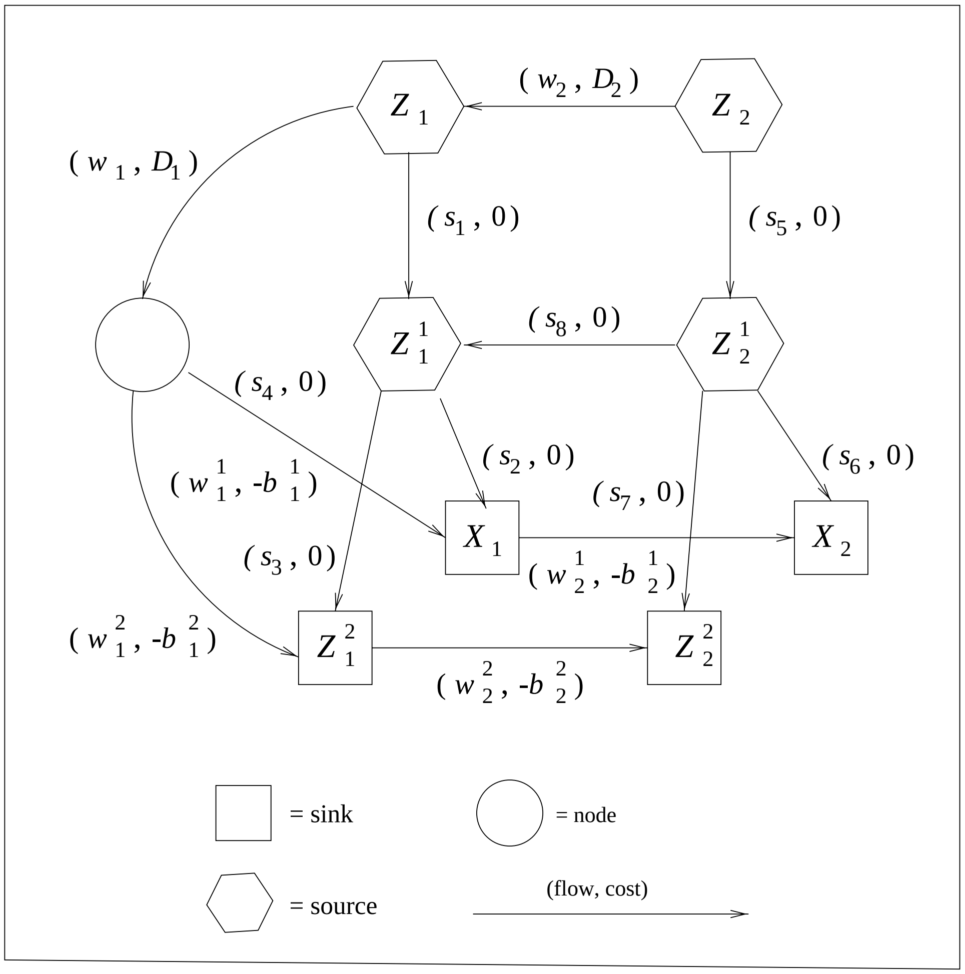

Figure 4.8: Dual network flow diagram for $S = 2$, $T = 2$

4.5 Computational Results and Implementation

The number of time periods, T, in SGH_2 should to be taken large enough so that all of the scheduled flights will be able to land. Specifically, T should satisfy $\sum_{t=1}^{T} \sum_{s=1}^{S} b_t^s \geq \sum_{t=1}^{T} \overline{D}_t$ or else there will be a surplus of $(Z_t + \sum_s Z_t^s)$ unresolved flights relegated to the time period $T + 1$ (the problem would still be feasible, however). In practice, the number of scheduled flights, $\overline{D}_t$, drops off dramatically in the night-time and early-morning hours (passengers tend to travel in the daytime) so one can assume that, in application, a planning horizon of 24 hours would be sufficient. Even if flights were delayed for extreme periods of time (longer than 24 hours) the airlines would probably divert the excess flights or cancel them altogether so that the total demand would be reduced to a level that could be accommodated in one 24-hour period. This means that although we take T to be finite, the planning horizon is effectively infinite.

The length of one time period, t, would probably be no smaller than say, 10 minutes.[2] Assuming that $T \leq 24$, an operational upper bound on T would be about 240. For a major international airport in the United States, a high AAR would be on the order of 60 or 70 flights per hour. Severe weather conditions could reduce this level to 30 or 40 flights per hour. GDP$'s$ are commonly formulated for a 4-6 hour duration.

We constructed three realistic instances of the SGH using the following guidelines. Each test case was solved for two levels of granularity of time periods, for

[2]There is little point to a further refinement of the time periods: there is inherent leeway in enroute travel times, departure times, gate availability, etc.

Test Case	Numb flights	Numb periods	Min per period	Opt Func Value	Time (sec)	Simplex Iteration	Node B&B
1a	624	12	60	642.00	0.07	51	0
1b	624	48	15	3329.00	0.30	267	0
2a	624	12	60	1088.00	0.07	59	0
2b	624	48	15	4209.00	0.37	289	0
3a	624	12	60	485.00	0.05	55	0
3b	624	48	15	1745.50	0.20	223	0

Table 4.1: SGH model performance

a total of six test cases. For each instance, we constructed three AAR scenarios. In test case 1, the AAR's were randomly generated numbers in the range 30 - 60. In test case 2, each AAR scenario drops from 60 flights per hour to 40 flights per hour for several hours, then returns to 60 flights per hour. The times were varied at which each AAR scenario drops and recovers. In test case 3, the first scenario is designed to be a fair weather scenario, the second is a bad weather scenario and the third is randomly generated.

In all instances, the costs used were 2.0 for ground holding and 5.0 for airborne holding. Each instance was solved using CPLEX 3.0 on a SPARC STATION 10.

The results in Table 4.1 show that the problem is highly tractable. The integer solution was obtained in zero nodes of the mixed integer program algorithm of CPLEX, meaning that the integer solution was obtained directly from the linear program relaxation. The largest number of iterations of the simplex procedure was 289 and the longest run time was barely more than half a second. Note that the run time is almost linear in the coarseness (length of) the time periods.

4.5.1 Sensitivity Analysis

Through experimentation, we have noticed a curious 'discrete' sensitivity in the optimal PAAR's (planned arrival acceptance rates) with respect to the cost ratio κ that causes us to question the wisdom of working with small number of scenarios. Specifically, upon input of AAR scenarios with bi-level patterns (to be defined shortly), the optimal PAAR's tend to closely match one of the AAR scenarios and remain unaltered as one raises κ, then, when certain critical points of κ are reached, the solution jumps to (closely follow) another scenario.

If, in working with this model, a single cost ratio κ can be arrived at, then this phenomenon is of little concern. However, if κ retains a certain degree of arbitrariness and the DM varies κ on a regular basis, then this phenomenon could be problematic. Suppose that in formulating a GDP, the DM wishes to fine-tune the conservativeness of a GDP by raising or lowering the value of κ just slightly. Then, if the values of κ are allowed to fluctuate around a critical point, then the model might output radically different solutions. We consider a lack of robustness an undesirable property so we now explore the underlying principle at work and propose a solution.

We say that an AAR scenario, $B^s = (b_1^s, b_2^s, ..., b_T^s)$,is **bi-level** if there are two capacity levels, $\beta_1^s \geq \beta_2^s$, and two time periods $t_1^s$and t_2^s (starting and ending, resp.) such that for every t, $t_1^s \leq t \leq t_2^s$ implies that $b_t^s = \beta_2^s$ and $b_t^s = \beta_1^s$ otherwise.

Then a bi-level AAR scenario is one that has a constant capacity for $(t_1^s - 1)$ time periods, drops to a lower capacity level at time t_1, then returns to the original level at time $t_2 + 1$. This pattern could arise in practice as a result of simplistic weather (and runway configuration) forecasting.

We now construct an instance of SGH_2 which displays this discrete behavior in extreme form. The lower portion of table 4.2 gives demand and three AAR scenarios for a fictitious airport over eight time intervals. Note the bi-level pattern in the scenarios and that the start times (and end times) of the scenarios coincide. The upper portion of table 4.2 gives the optimal PAAR's (generated by CPLEX) for values of κ between 1.2 and 4.0, taken in increments of 0.2. The probabilities of the scenarios are evenly distributed at $p(s) = 1/3$, for $s = 1, 2, 3$. The phenomenon to be observed is that there are only three optimal solutions, corresponding to the AAR scenarios, respectively, and that the solution jumps from one scenario to another at two critical values of κ at around 3.0 and 1.4.

For this artificial example, the critical points can be exactly computed by deducing the optimal PAAR's for each value of κ. Since demand is constant at 70 flights per time interval and since capacity in each scenario is 70 for $t = 1, 2, 7, 8$, one should accept 70 flights in each of these time intervals to avoid airborne or ground delay. Then in any optimal solution, for any value of κ, we will have that $X_1 = X_2 = X_7 = X_8 = 70$ and each solution will be characterized strictly by the optimal value of X_t for $t = 3, 4, 5, 6$. (Note: The optimality of the values of X_1, X_2, X_7, X_8 requires a more rigorous argument but would be tangental to our current discussion.)

We will assume that ground-holding cost, $\tilde{c}$, is held constant at $\tilde{c} = 1.0$ so that $\kappa = \widehat{c}/\tilde{c} = \widehat{c}$ and a change in κ is simply a change in the airborne holding cost, $\widehat{c}$. Suppose that $3.0 \leq \kappa \leq 4.0$. For $t = 3, 4, 5, 6$, the capacity in each scenario is at least 30 so there is no cost incurred in accepting at least 30 flights in each time period, i.e., $X_t \geq 30$ for $t = 3, 4, 5, 6$. If we increase the value of X_t beyond 30 for any such t, then capacity will be exceeded in scenario 1 and one will incur an

cost ratio	**t=1**	**t=2**	**t=3**	**t=4**	**t=5**	**t=6**	**t=7**	**t=8**
1.2	70	70	65	65	65	65	70	70
1.4	70	70	65	65	65	65	70	70
1.6	70	70	50	50	50	50	70	70
1.8	70	70	50	50	50	50	70	70
2	70	70	50	50	50	50	70	70
2.2	70	70	50	50	50	50	70	70
2.4	70	70	50	50	50	50	70	70
2.6	70	70	50	50	50	50	70	70
2.8	70	70	50	50	50	50	70	70
3	70	70	50	50	50	50	70	70
3.2	70	70	30	30	30	30	70	70
3.4	70	70	30	30	30	30	70	70
3.6	70	70	30	30	30	30	70	70
3.8	70	70	30	30	30	30	70	70
4	70	70	30	30	30	30	70	70
demand	70	70	70	70	70	70	70	70
scen 1	70	70	30	30	30	30	70	70
scen 2	70	70	50	50	50	50	70	70
scen 3	70	70	65	65	65	65	70	70

Table 4.2: Optimal solutions as a function of cost ratio

expected airborne holding cost ($EAHC$) given by

$$EAHC = p(1) \cdot \widehat{c} = (1/3) \cdot \kappa.$$

For $3.0 \leq \kappa \leq 4.0$, we have that $EAHC \geq \widetilde{c} = 1.0$, and it is cheaper (in expected value) to hold subsequent flights on the ground and no more flights should be accepted. This fixes the optimal solution at $X_t = 30$ for $t = 3, 4, 5, 6$ (i.e., scenario 1).

But when $EAHC \leq \widetilde{c}$, the values of X_t will change. This establishes our first critical point κ at $EAHC = \widetilde{c} = 1.0$ or, in other words,

$$\begin{aligned} EAHC &= p(1) \cdot \widehat{c} = (1/3) \cdot \kappa = 1.0 \\ \kappa &= 3.0\ . \end{aligned}$$

For $1.6 \leq \kappa \leq 3.0$, we find that $EAHC \leq \widetilde{c}$. In time periods $t = 3, 4, 5, 6$, it becomes cheaper to accept additional flights and their subsequent (expected) airborne holding costs. We increment the integer values of X_t beyond 30 to find the optimal solution. But when $X_t > 50$, capacity in scenario 2 is exceeded as well as in scenario 1. The new $EAHC$ calculation for any flight beyond the 50^{th} flight is

$$EAHC = [p(1) + p(2)] \cdot \widehat{c} = [1/3 + 1/3] \cdot \widehat{c} = [2/3] \cdot \widehat{c}$$

Since $1.6 \leq \kappa \leq 3.0$, $EAHC \geq \widetilde{c}$ and ground holding is preferable. This fixes the optimal solution at $X_t = 50$ for $t = 3, 4, 5, 6$, (scenario 2). This will hold until we reach our second critical point κ at $EAHC = \widetilde{c}$ or,

$$\begin{aligned} [2/3] \cdot \widehat{c} &= \widetilde{c} \\ 2/3 &= 1/\kappa \\ \kappa &= 1.5 \end{aligned}$$

For values of κ such that $\kappa < 1.5$, the optimal solution switches to $X_t = 65$, for $t = 3, 4, 5, 6$, (i.e., scenario 3). We do not consider values of $\kappa \leq 1.0$ because there is no need for this model when airborne holding is cheaper than ground holding. Reflection upon the above calculations shows that the precise values of the critical points is heavily dependent upon the probabilities, $p(s)$. Therefore, when working with a value of κ near a critical point, the solution is also highly sensitive toward a redistribution of the probabilities.

The phenomenon of critical points in the cost ratio begins to erode if the start and end times of the AAR scenarios are staggered as in the bottom of table 4.3. The results at the top of table 4.3 are similar to but not as marked as the previous example. Again, the optimal solution changes significantly at $\kappa = 1.5$ and $\kappa = 3.0$ but there are two major differences this time. First, the majority of the optimal solutions do not exactly agree with any of the scenarios. Second, note that $\kappa = 1.5$ affects time intervals $t = 3, 4, 5$ while $\kappa = 3.0$ affects $t = 4, 5, 6, 7$. In this respect, the concept of a critical point becomes heavily dependent upon the time interval, hence, begins to lose its meaning.

At the other extreme, when the AAR scenarios rise and fall erratically, critical points vanish altogether and the model becomes robust with respect to perturbations of the cost ratio, κ, and redistributions of the probabilities.

If, in practice, the only AAR scenarios that can be forecasted tend to display bi-level patterns and, worse yet, tend to coincide in start and end times of their reduced capacity, then the optimal solution will be very stable for some ranges of cost ratios and highly unstable for others.

One way to guard against this behavior is to work with a large number of AAR scenarios. Another alternative is to refine the probability distributions. Each sce-

cost ratio	**t=1**	**t=2**	**t=3**	**t=4**	**t=5**	**t=6**	**t=7**	**t=8**
1.2	70	70	70	65	65	65	70	70
1.4	70	70	70	65	65	65	70	70
1.6	70	70	30	50	55	65	70	70
1.8	70	70	30	50	55	65	70	70
2	70	70	30	50	55	65	70	70
2.2	70	70	30	50	55	65	70	70
2.4	70	70	30	50	55	65	70	70
2.6	70	70	30	50	50	65	70	70
2.8	70	70	30	50	50	65	70	70
3	70	70	30	50	50	65	70	70
3.2	70	70	30	30	30	30	30	70
3.4	70	70	30	30	30	30	30	70
3.6	70	70	30	30	30	30	30	70
3.8	70	70	30	30	30	30	30	70
4	70	70	30	30	30	30	30	70
demand	70	70	70	70	70	70	70	70
scen 1	70	70	30	30	30	30	30	70
scen 2	70	50	50	50	50	70	70	70
scen 3	70	70	70	65	65	65	70	70

Table 4.3: The erosion of critical points

narios, s, could be converted to a set of 'micro-scenarios', $s_1, s_2, ..., s_n$, such that $\sum p(s_i) = p(s)$. For each time period, t, the capacities of these micro-scenarios could be designed cover a gradient of values hovering around the capacity of scenario s.

Chapter 5

Closing Remarks

This dissertation makes three major contributions. First, it provides practical solutions to problems in air traffic flow management. Secondly, it adds to the knowledge base of integer programming by providing theoretical and computational results. Third, it provides guidance for research in air traffic by presenting in a precise manner the latest methodologies and paradigms in air traffic flow management developed by the Collaborative Decision Making (CDM) work group.

The stochastic ground-holding model presented in this dissertation finds the optimal trade-off between airborne-holding and ground-holding in the formulation of a ground delay program. By treating flights on an aggregate level, this integer programming model avoids the need to consider the assignment of every flight to each time period in the planning horizon. This yields a simpler, more efficient solution, hence, an improvement over the previous models found in the literature. More importantly, the model is designed to be an integral component of the collaborative decision-making process and can be easily integrated into the FSM decision support tool. This makes it available for immediate use.

A particularly powerful model was developed to find the minimum cost solution to the assignment of arrival times to flights bound for a single airport while holding together banks of flights. By producing facet-inducing constraints, this model greatly cuts down on the computational resources necessary to solve this difficult integer programming problem. It was demonstrated that realistic instances of the problem can be solved in just minutes using a commercial solver on a personal computer or ordinary work station.

The CDM working group is an extremely qualified and influential team of aviation experts. Their impact promises to be dramatic. As ATFM (air traffic flow management) moves toward an increasingly collaborative setting, the successful contribution of researchers to ATFM will be heavily dependent upon their understanding of CDM goals and the type of decision support tools required by decision makers in all areas of the aviation community. This dissertation has made a significant contribution to this understanding.

The development of the models in this dissertation has introduced and resolved several intriguing theoretical issues in integer programming and combinatorial optimization. In the work on stochastic ground-holding, it was shown that the flow model introduced is non-network in the primal but that the dual can be transformed into a network flow problem. This renders it a powerful tool for obtaining rapid solutions to more general types of stochastic programming problems. In the work done on the ground-holding problem with banking constraints, it was shown that under mild assumptions each of the constraints generated by two of the models represents a facet of the convex hull of the set of integer solutions. These results carry over to a job-scheduling generalization of the problem.

The search for an efficient solution to the ground-holding problem with banking constraints showed the enormous benefits that can be obtained through the exploration of alternative formulations. Computational analyses showed that one of the models was far superior to the others in performance and a polyhedral projection was used to show that another one of the models is equivalent in LP strength. The development of the models employed a number of key techniques related to the formulation of integer programs such as the introduction of auxiliary variables and the selective imposition of integer restrictions.

This dissertation paves the way for several research topics. For instance, the banking constraint models presented here could be modified for integration with a decision support tool such as the flight schedule monitor (FSM), currently used CDM. The axiom of GHB that each bank must arrive within its specified time window could be reconsidered. An alternative model might allow for the temporal expansion of a bank beyond the desired parameter but at a penalty reflected in the objective function. The challenge there would be to find a concise mathematical representation of the expansion. Another model could exempt a limited number of flights from the banking constraints, thus allowing for a trade-off between banking and reduced overall delay costs. Such models could be modified so that their behavior closely mimics the decision making processes of the airlines and the FAA in the formulation of a ground delay program.

The stochastic flow model introduced in this dissertation assumes the production of multiple arrival acceptance rate scenarios with associated probabilities. The generation of the scenarios and their probabilities represents unique research challenges. This requires the fusion of weather forecasts with optimal runway configurations using a combination of statistical techniques and analysis of the impact of meteorological conditions on incoming aircraft. Moreover, the interaction of incoming aircraft with departing aircraft could lead to many interesting problems in optimization.

During the coming years, the growth of air traffic in the NAS (National Airspace System) is inevitable. Crowded conditions in the NAS will call for more efficient use of resources. Collaborative decision making will call for the development of more sophisticated decision support tools. As tracking capabilities and communication systems grow, the demand for innovative models will increase dramatically. The concept of free flight alone will pose a myriad of technical challenges. All of these factors will only heighten the role of integer programming and combinatorial optimization in the future of air traffic flow management.

Appendix A

RBS and Compression Algorithms

In this Appendix, formal presentations of the RBS (ration by schedule) and Compression algorithms are presented as well as more in-depth discussions that would have detracted from their initial presentation. For an overview of the algorithms as well as their scope and purpose, the reader is referred to the abbreviated versions in Section 2.2.2

A.1 Input to algorithms:

- A set F of flights and a partition Φ of F into airlines
- For each $f \in F$, the following data fields

  $AIR_f \in \Phi$ =airline of flight f

  $ETA_f,\ ETD_f$ =estimated arrival/departure times for f

  $CTA_f,\ CTD_f$ =controlled times of arrival/departure for f

  $EDCT_f$ =estimated departure clearance time of f

  $OGTA_f,\ OGTD$ =original gate time of arrival/departure of f

  (in general, $O-$ prefix implies an original time e.g., $OETA_f$)

- Time parameters $current\_time,\ start\_time = t_0,\ end\_time = t_1, Taxi\ (time)$
- Contiguous time slots $S = t_0, t_0 + 1, ..., t_1$ (created in RBS, input to Compression)
- For each $t \in S$, the following fields:

  $flight\,(t) \in F \cup \{null\}$

  $owner\,(t) \in \Phi \cup \{null\}$

  $status\,(t) \in \{open,\ released,\ filled, hold\}$

A.1.1 The RBS Algorithm

1. Find the set of flights, $I \subseteq F$, to be included in the program. Let I = $A \bigcup B$, where

$$\begin{aligned} A &= \{f \in F : t_0 \leq ETA_f \leq t_1\} \\ B &= \{f \in F : t_1 < ETA_f \quad and \quad (t_0 \leq CTA_f \leq t_1\ \ or\ \ t_0 \leq OGTA_f - Taxi \leq t_1)\} \end{aligned}$$

2. Find the set of flights, $E \subseteq I$, to be exempted from (delay within) the program.

   This set is user specified.

3. Split non-exempted (but included) flights $I - E$ into two disjoint sets, F_1 and F_2, defined

   via

$$\begin{aligned} F_1 &: = \{f \in I - E : f\ has\ been\ assigned\ a\ CTA\ at\ least\ once\} \\ F_2 &: = (I - E) - F_1 \end{aligned}$$

Note: $I = E \smile F_1 \smile F_2$.

4. Compute *earliest*_CTA for each $f \in I$

   if $f \in E$,

$$earliest\_CTA := \begin{cases} ETA_f \ , \ if \ f \ has \ a \ slot \ ID \\ min\left(ETA_f, \ OGTA - Taxi\right), \ else \end{cases}$$

   if $f \in F_1 \smile F_2$,

$$earliest\_CTA := \begin{cases} OGTA - Taxi, & if \ current\_time \leq OETD_f \\ current\_time + p + \left(OETA_f - OETD_f\right), & otherwise \end{cases}$$

   Note: p is a positive parameter, user-specified

5. Create queues of flights.

   (a) $Q(E) := q_1^E, q_2^E, ..., q_{|E|}^E$, from E using ETA as a priority.

   (b) $Q(F_1) := q_1^{F_1}, q_2^{F_1}, ..., q_{|F_1|}^{F_1}$, from F_1 using CTA as a priority.

   (c) $Q(F_2) := q_1^{F_2}, q_2^{F_2}, ..., q_{|F_2|}^{F_2}$,from F_2 using $(OGTA - Taxi)$ as a priority.

   (d) $Q := q_1^E, q_2^E, ..., q_{|E|}^E, \quad q_1^{F_1}, q_2^{F_1}, ..., q_{|F_1|}^{F_1}, \quad q_1^{F_2}, q_2^{F_2}, ..., q_{|F_2|}^{F_2}$

   Re-index via $Q = q_1, q_2, ..., q_{|I|}$. Note that $\bigcup_{k=1}^{|I|} q_k = I$.

6. Create the set S of virtual slots according to user-specified AAR's.

   For instance, if AAR= 6 *flights per hour*, $t_0 = 1801$, $t_1 = 2159$,

   then $S = \{1810, 1820, ..., 2150, 2200\}$. When AAR> 60, use suffixes to mark

   subdivisions of minute intervals, e.g., $1801A$, $1801B, ...$

```
7. Assign a CTA and $OCTA$ (if necessary) to each $f \in I$
    Let $Q$ be as in Step 5(d) and let $S$ be as in Step 6.
    for $f = q_1, q_2, \ldots, q_{|I|}$,
        $Let\ t' := min\,(t \in S : t \geq earliest\_CTA_f\ and\ status\,(t) = open)$
        $CTA_f := t'$
        $CTD_f := CTA_f - OETA_f + OETD_f\,,\ if \in F_1 \cup F_2$
        $EDCT_f := \begin{cases} OETD + 1,\ if\ f \in E \\ CTD_f\,,\ if\ \ f \in F_1 \cup F_2 \end{cases}$
        $OCTA_f := CTA_f\ ,\ OCTD_f := CTD_f\ ,\ if\ \ f \in F_2$
        $ETA_f := CTA_f\,,\ if\ f \in F_1 \cup F_2$
        $flight\,(t') := f$
        $owner\,(t') := AIR_f$
        $status\,(t') := \begin{cases} open,\ if\ f \in E\ and\ CTA_f < ETA_f \\ filled,\ else \end{cases}$
        $S := S - \{t'\}$
    end for
```

end RBS algorithm.

The RBS algorithm is surprisingly complex considering that it is intrinsically a fist-scheduled, first-served algorithm. Since RBS is to be implemented prior to the Compression algorithm or the cancellation/substitution process, it carries with it the added burden of establishing those flights both in the program and non-exempted. This is the function of steps 1 and 2. Step 3 allows for the possibility that RBS or compression have been run once before. Therefore, it is arguable that steps 1 - 3 are preliminary to the entire GDP formulation procedure and that RBS truly begins in step 4.

Note that in step 1, there are three sufficient conditions for a flight f to be included in the program: (1) ETA_f falls within the program horizon or (2) ETA_f is beyond the program horizon but CTA_f falls within the program horizon or (3) ETA_f is beyond the program horizon but the original scheduled arrival time of f falls within the program horizon. The latter of the three ensures that an airline will not be penalized for reporting a delay.

The user-specified exemptions of step 2 will be discussed more in Chapter 4. In step 4, the status of an exempt versus non-exempt flight is acknowledged.

Step 5 is the heart of the algorithm. It rations slots by the following priority scheme: exempt flights first, flights already assigned a CTA second, and, lastly, non-exempt flights that require a CTA. Steps 6 could be considered preliminary and step 7 is a necessary labeling system for future runs of RBS.

A.1.2 The Compression Algorithm

main Algorithm (Compression)

```
1. for each slot $t = t_1, t_2, ..., t_{|S|}$
        if status($t_k$) = $filled$ or $hold$
            go to $t_{k+1}$
        else        [$status = open$ or $released$]
            call subroutine $fill\_slot$ ($t$)
```

```
end main algorithm

subroutine $fill\_slot\,(t)$        [try to fill slot $t$ with a flight]

1. if $status\,(t) = open$ then

    (a) form one queue $Q$ of all flights ordered by CTA
    (b) call subroutine $search\,(Q,\, t)$

1. if $status\,(t) = released$, then

2. let $A := owner\,(t)$

    (a) form two queues
          $Q^A$ := flights from airline $A$ ordered by CTA
          $Q^{\overline{A}}$ := flights not in airline $A$ ordered by CTA
    (b) call subroutine $search\left(Q^A, t\right)$
    (c) if $search\left(Q^A, t\right)$ returns $no\_flight\_found$, then
          call $search\left(Q^{\overline{A}}, t\right)$
    (d) if $search\left(Q^A, t\right)$ returns $hold\_slot$, then end subroutine
end subroutine $fill\_slot\,()$

subroutine $search\,(Q,\, t)$        [find a flight in $Q$ to assign to $t$]

1. input queue $Q$ and time slot $t$

2. if $Q = \phi$ then return $hold\_slot$ and end subroutine

3. repeat

        (a) $f$ := next flight in $Q$
    (b) if $feasible(f,t) = true$ then
        $old\_owner := owner(t)$          [save info on $t$]
        $old\_status := status(t)$
        $t' =$CTA$_f$                     [next slot to consider]
        CTA$_f := t$                      [assign $f$ to $t$]
        $status(t) := filled$
        $owner(t) := Air_f$
        (c) if $old\_status = open$ then
            $status(t') := open$
        (d) if $old\_status = released$ then
            $status(t') := released$
            $owner(t') := old\_owner$
            $fill\_slot(t')$
    (e) $Q := Q - \{f\}$
  until  $status\,(t) = filled$ or $Q = \phi$
4. if $status\,(t) = filled$ then return $flight\_found$ else return $no\_flight\_found$
  **end subroutine** $search\,(\ ,\ )$

  **subroutine** $feasible\,(f,t)$
5. input flight $f$, time slot $t$
6. check that $f$ meets feasibility criteria (user specified) for assigning $f$ to $t$
```

```
7. if $f$ feasible for $t$ then return $true$ else return $false$
   end subroutine $feasible$( , )
```

The main algorithm of Compression scrolls through the time slots in ascending order and attempts to fill available slots by calling the subroutine *fill_slot*( ). This subroutine sorts slots with an *open* status from those with a *released* status. If a slot t is *released*, then it has a controlling airline, A, and, in step 1(b), the subroutine *search* (,) attempts to fill t with the next available flight from A. If this is not possible, then in step 1(c), the subroutine *search* (,) attempts to fill t with the next available flight not in airline t. If a slot is *open*, *search* ( ) is called in step 2 on the entire collection of flights. Note that under any circumstances, flights are considered in order of increasing CTA [see steps 1(a) and 2(a)].

It is quite possible for the subroutine *search* ( ) to fail in the attempt to assign a flight to a time slot from the input queue. Moreover, the entire algorithm may leave a slot with no flight assigned to it. This is a desirable feature, since a slot may be too early for any flight in the program.

One of the key aspects of this algorithm is that the status of these unfilled slots is unaltered. Suppose that an *open* slot t cannot be filled with a flight from the controlling airline, AL, because it has no flights below slot t. Then the subroutine *search* ( ) returns a *hold_slot* status in step 2, indicating that the input queue of flights from AL is empty, and by step 1(d) of *fill_slot*( ), the attempt to fill slot t is halted. If the ground delay program is extended, AL might suddenly have a flight that can be moved into t. By retaining the *open* (but not *released*) status of t, AL can make future use of the resource.

The upward movement of flights from a single airline (substitution stream) is ignited by step 3(c). Suppose that a slot t has the *open* status and owner AL. If a flight f is moved into t from a (lower) slot t', then ownership of slot t is transferred to t', regardless of the airline that owns f, and the subroutine $fill\_slot(\,)$ is recursively called on t'. A flight from AL is then sought to for t'. If f belongs to some airline $BL \neq AL$, then f is said to have acted as a *bridge* for the substitution stream. Note that both AL and BL have profited from the vacancy created at slot t by airline AL.

In the interest of NAS user compliance, a more refined version of Compression is used in which flights are designated as Class I or Class II. Class II flights are those created within 48 hours of the current time. If a Class II flight is cancelled from a slot t, then the status of t is declared to be *released*, thus blocking the controlling airline from creating a substitution stream. This is intended to discourage the airlines from scheduling dummy flights just prior to the program.

The feasibility criteria in the subroutine $feasible\,(\,)$ has deliberately been left vague. There are several possibilities for this. The most direct is that a flight f can be moved into a slot t only if $ETA_f \geq t$. More generally, the requirement is that $ETA_f \geq t+k$, where $k \geq 0$ is a fixed parameter to prevent minor movements of flights. Also, feasibility of assignment should vary with the current time. The airlines require a minimum notification if a flight is to be moved earlier than its current CTA. This minimum requirement is currently set at 30 minutes for all airlines.

Example of Compression

Consider a GDP implemented between times 1100 and 1310, with an AAR of six flights per hour (the numbers are simplistic for the purposes of illustration). The last seven time slots of the schedule are shown below.

Before Cancellation/Substitution			
slot	status	owner	flight-earliest_CTA
1201-1210	*filled*	A	A100-1149
1211-1220	*filled*	B	B100-1151
1221-1230	*filled*	C	C100-1155
1231-1240	*filled*	A	A200-1228
1241-1250	*filled*	B	B200-1216
1251-1300	*filled*	A	A300-1235
1301-1310	*filled*	D	D100-1335

The airlines are now free to make cancellations and substitutions within their own time slots. Suppose that after this process, airline A has cancelled flight A100 or moved it to a higher position in the schedule, thus vacating slot 1201. Similarly, airline B has cancelled or moved up B100, thus vacating slot 1211.

After Cancellation/Substitution			
slot	status	owner	flight-earliest_CTA
1201-1210	*released*	A	-
1211-1220	*released*	B	-
1221-1230	*filled*	C	C100-1155
1231-1240	*filled*	A	A200-1228
1241-1250	*filled*	B	B200-1216
1251-1300	*filled*	A	A300-1235
1301-1310	*filled*	D	D100-1335

As it stands, airline A cannot make use of the 1201 time slot because the earliest_CTA's of flights A200 and A300 are too early for the 1201 time slot. Airline B could have moved B200 into the 1211 slot but is trusting the compression algorithm to do this on their behalf. Note that airlines A and B have retained ownership of their respective, vacated slots.

The main algorithm of Compression calls upon subroutine *fill_slot*( ) to fill slot 1201 with the next available flight of airline A. Subroutine *search*( ) reports that none of the A flights are feasible for this slot (note the earliest_CTA's of A200 and A300) so *search*( ) is called upon to fill slot 1201 with the next available flight from the other airlines. These flights are considered in the order of CTA, which is the same as the positioning in the schedule above. C100 is the first available, so it is moved into slot 1201, as below.

Flight C100 moved up			
slot	status	owner	flight-earliest_CTA
1201-1210	*filled*	C	C100-1155
1211-1220	*released*	B	-
1221-1230	*released*	A	-
1231-1240	*filled*	A	A200-1228
1241-1250	*filled*	B	B200-1216
1251-1300	*filled*	A	A300-1235
1301-1310	*filled*	D	D100-1335

Step 5(c) of *search*( ) changes ownership of slot 1221 (vacated by C100) from C to A and marks it as released. This is crucial to the remainder of the algorithm. Although airline A could not use slot 1201 directly, compensation will be made to A by moving one of it's flights as close as possible to slot 1201. In particular, *fill_slot*( ) is called on slot 1221 and the attempt is made to move an A flight into this slot. Subroutine *search*( ) recognizes that the next flight, A200, in the queue of A flights (ordered by CTA) is feasible to the 1221 slot and places it there. The vacated slot, 1231, is marked with type A ownership and *fill_slot*( ) is called on 1231. This moves flight A300 into slot 1231 and the schedule is now as below.

End of Substitution Stream of A flights			
slot	status	owner	flight-earliest_CTA
1201-1210	*filled*	C	C100-1155
1211-1220	*released*	B	-
1221-1230	*filled*	A	A200-1228
1231-1240	*filled*	A	A300-1235
1241-1250	*filled*	B	B200-1216
1251-1300	*hold*	A	-
1301-1310	*filled*	D	D100-1335

Note that *search*( ) has marked slot 1251 with *hold* status. This marks the end of the substitution stream created by the vacancy at slot 1201 and prevents flight D100 from being moved into slot 1251, even though D100 is feasible for it. If the program is extended beyond 1310, airline A will have the opportunity to move one of its flights up into the 1251 time slot. The *for* loop of the main algorithm calls upon *fill_slot*( ) to fill the 1211 slot with B200 and the algorithm terminates with the results below.

End of Compression			
slot	status	owner	flight-earliest_CTA
1201-1210	*filled*	C	C100-1155
1211-1220	*filled*	B	B200-1216
1221-1230	*filled*	A	A200-1228
1231-1240	*filled*	A	A300-1235
1241-1250	*hold*	B	-
1251-1300	*hold*	A	-
1301-1310	*filled*	D	D100-1335

The net result of the Compression algorithm was to push flights up in the schedule and force the unusable slots to the bottom of the schedule, thus minimizing overall delay. Airline A was compensated for slot 1201 even though it could not use it and a direct substitution was made for airline B. All airlines except C received a reduction in delay.

Appendix B

Proofs

B.1 Proof of Lemma 1

The algorithm below produces n linearly independent, linear combinations of the vectors in S, where $n = (FT - F)$ and S is the set of integer solutions in GHB_1^C

.

```
Algorithm 1
Note: Let $w = w_b$ , for ease of notation.
STEP 1:
For $j = 1, 2, ..., T$
    Set vector $Y$ via:
      Block 1: $Y_{1,j} = 1 \qquad Y_{1,k} = 0, k \neq j$
      if $1 \leq j \leq (Tw + 1)$
        Block 2: $Y_{2,j+w-1} = 1$ $Y_{2,k} = 0,\ k \neq j + w$
      else
        Block 2: $Y_{2,T} = 1 \qquad Y_{2,k} = 0, k \neq T$
      end if
      Block $p > 2$ : [Set in any feasible manner]
      Output row vector $U = Y$ (Note: $U_{1,j} = 1, U_{1,k} = 0$ for all $k < j$ )
  end for
  STEP 2:
  For $j = 1, 2, ..., (T-2)$
    Set vectors $X$ and $Y$ via:
    if $1 \neq j \neq (w-2)$
      Block 1: $Y_{1,1} = 1, X_{1,1} = 1$      $Y_{1,k} = X_1,\ k = 0,\ k \neq j$
      Block 2: $Y_{2,j} = 1, X_{2,j+1} = 1$    $Y_{2,k} = 0$ for $k \neq j$, $X_{2,k} = 0$ for $k \neq j+1$
    if $(w_b - 1) \leq j \leq (w-1) + t - 1$
      $u = (j - w + 2)$
      Block 1: $Y_{1,u} = 1\ X_{1,u} = 1\ Y_{1,k} = X_{1,k} = 0,\ k \neq u$
     Block 2: $Y_{2,j} = 1\ X_{2,j+1} = 1\ Y_{2,k} = 0\ for\ k \neq j\ and\ X_{2,k} = 0\ for\ k \neq j+1$
    $if (w + t - 1) \leq j \leq (T-2)$
      Block 1:$Y_{1,j+1} = 1\ X_{1,j+1} = 1\ Y_{1,k} = X_{1,k} = 0, k \neq j+1$
       Block 2:$Y_{2,j+1} = 1\ X_{2,j+2} = 1\ Y_{2,k} = 0\ for\ k \neq j+1\ and\ X_{2,k} = 0\ for\ k \neq j+2$
  end if
      Block $p > 2$: $Y_{p,k} = X_{p,k}$ for all $k$. [Set in any feasible manner]
      OUTPUT $Z = (Y - X)$ (Note: $U_{2,j} = 1, U_{2,k} = 0$ for all $k < j$)
   end for
   Repeat STEP 3 for each block $m = 3, 4, ..., F$
  STEP 3:
   For $j = 1, 2, ..., (T-1)$
      Set vectors $X$ and $Y$ via:
      if $1 \leq j \leq (T - w + 1)$
```

```
        Block 1: $Y_{1,j} = 1$ $X_{1,j} = 1$        $Y_{1,k} = X_{1,k} = 0$,else.
        Block 2: $Y_{2,(j+w-1)} = 1$ $X_{2,j+w-1} = 1$ $Y_{2,k} = X_{2,k} = 0$, else.
        Block $m$: $Y_{m,j} = 1$ $X_{m,j+1} = 1$     $Y_{m,k} = X_{m,k} = 0$, else.
      else
        Block 1: $Y_{1,j} = 1$ $X_{1,j} = 1$        $Y_{1,k} = X_{1,k} = 0$, for $k \neq j$
        Block 2: $Y_{2,T} = 1$ $X_{2,T} = 1$        $Y_{2,k} = X_{2,k} = 0$ for $k \neq T$
        Block $m$: $Y_{m,j} = 1 X_{m,j+1} = 1$     $Ym, k = 0$ for $k \neq j$ and $X_{m,k} = 0$ for
$k \neq j+1$
        Block $p > 2, \neq m$: $Y_{p,k} = X_{p,k}$ for all $k$. [Set in any feasible manner]
      Output row vector $U = (Y - X)$.
    end for
     Block $p$: $Y_{p,k} = X_{p,k}$ for all $k$. [Set in any feasible manner]
      Output row vector $U = (Y - X)$.
  end for
```

end Algorithm 1

Proof that Algorithm 1 is correct:

Form a matrix, A, by letting the k^{th} row of matrix A be the k^{th} (row) vector output by Algorithm 1. Note that A has FT columns. To show that the algorithm is correct, it will suffice to show that the rows of A are linearly independent, linear combinations of vectors from S, where S is the set of integer solutions in GHB_1^C . To this end, we will show three things:

(i) The number of rows in A is $n = (FT - F)$

(ii) The rows of A are linearly independent

(iii) Each row of A is a linear combination of vectors from S

Proof of (i) : Step 1 yields T vectors. Step 2 yields $(T - 2)$ vectors. Step 3 yields $(T-1)$ vectors for each of its $(F-2)$-many executions. The total number of vectors output by the algorithm is given by:

$$T + (T-2) + (F-2)(T-1) = F(T-1) = FT - F = n$$

The fact that these vectors are distinct will follow from the linear independence of the vectors.

Proof of (ii): To show that the rows of A are linearly independent, it will suffice to show that A is in row-echelon form and that each row is a pivot row. By construction, every (row) vector has a lead "1" in component (i, j), for some (i, j). We will show that the lead entries in the rows of matrix A are staggered, left to right. Let U be any vector output by the algorithm except the last. Suppose that the lead entry of U occurs in the position (i, j) (i.e., $U_{ij} = 1$ and for all $m < j$, $U_{i,m} = 0$ and for all $k < i$ and all h, $U_{kh} = 0$). The lead-entry of the next vector, U^*, will occur either in the same block i, and the position $(i, j+1)$, (whenever U^* is created in the same for-loop) or it will occur in the next block, $(i+1)$ (whenever U^* is created in the subsequent for-loop). In either case, the lead-entry of U^* is strictly to the right of the lead entry in U. So, A is in row-echelon form and each of its n rows is a pivot row.

Proof of (iii): Lastly, we must show that each row of A is a linear combination of vectors in S. Each vector, U, output by the algorithm is formed by either $U = (Y - X)$ or $U = Y$, so, clearly, U is a linear combination of the vectors X and Y. Next, we must show that X and Y are in S. That is, we must show that X and Y are

(a) integer vectors

(b) solutions to GHB and

(c) meet constraint C at equality

Since the components of X and Y are binary, (a) is clear. Over each block, the components of Y (and X) sum to one, so constraints (1.3) are satisfied. Let j be $1 \leq j \leq T$. The number of k for which $Y_{k,j} = 1$ (or $X_{k,j} = 1$) is less than or equal to two (except for $j = T$) and since we have assumed that for all t, b_t 2, Y (and X) satisfies the capacity constraints. Let $Y_{i,j} = 1$ and $Ym, n = 1$ where $m \neq i$. By construction, $|j - n| \leq w_b$, so Y satisfies the banking constraints. The same holds for X. So, X and Y are solutions to GHB, and (b) is shown. Finally, to show (c), note that for any vector, X, constraint C reads

$$\sum_{s=1}^{t} X_{1,s} + \sum_{s=t+w_b}^{T} X_{2,s} \leq 1.$$

To show that X meets C at equality, it suffices to show that exactly one of the following holds true:

(i) $X_{1,s} = 1$ for exactly one $s \in \{1, 2, ..., t\}$

(ii) $X_{2,s} = 1$ for exactly one $s \in \{t + wb, t + wb + 1, ..., T\}$

By this technique, we will show that, at each step of the algorithm, both X and Y meet C at equality.

Let $w = w_b$.

```
STEP 1:
for $j = 1, 2, ..., t$,
   $Y_{1,j} = 1 \Rightarrow$ (i) true for $Y$
   $Y_{2,j+w-1} = 1 \Rightarrow Y_{2,k} = 0$ for all $k \geq t + w$ (ii) false for $Y$
for $j = (t + 1), (t + 2), ..., T$
   $Y_{1,j} = 1 \Rightarrow Y_{i,k} = 0$ for all $k \leq t \Rightarrow$(i) false for $Y$
   $Y_{2,T} = 1 \Rightarrow (ii)$ true for $Y$
STEP 2:
for $j = 1, 2, ..., w - 2 + t$
```

$Y_{1,j} = 1$ for some $1 \leq j \leq t \Rightarrow$ (i) true for Y

$Y_{2,j} = 1 \Rightarrow Y_{2,k} = 0$ for all $k \geq t + w$ (ii) false for Y

$X_{1,j} = 1$ for some $1 \leq j \leq t$ (i) true for X

$X_{2,j} + 1 = 1 \Rightarrow X_{2,k} = 0$ for all $k \geq t + w \Rightarrow$(ii) false for X

for $j = (w + t), (w + t) + 1, ..., (T - 1)$

$Y_{1,j} = 1$ (i) false for Y

$Y_{2,u} = 1$ for $u \geq w + t \Rightarrow$(ii) true for Y

$X_{1,j} = 1 \Rightarrow$ (i) false for X

$X_{2,u} = 1$ for $u \geq w + t \Rightarrow$ (ii) true for X

STEP 3:

Note that $Y_{1,k} = X_{1,k}$ and $Y_{2,k} = X_{2,k}$ for all k

for $j = 1, 2, ...t$

$Y_{1,j} = 1 \Rightarrow$ (ii) true for Y, (i) true for X

$Y_{2,j+w-1} = 1 \Rightarrow Y_{2,k} = 0$ for all $k \geq t + w \Rightarrow$ (ii) false for Y, (ii) false for Y

for $j = t, t + 1, ..., T$

$Y_{1,j} = 1 \Rightarrow$ (i) false for Y, (i) false for X

$Y_{2,u} = 1$ for some $u \geq w + t \Rightarrow$ (ii) true for Y, (ii) true for X

Thus, each X and each Y produced by algorithm 1 satisfy C at equality. In all, we have shown that there are (at least) n linearly independent (hence, affinely independent), integer vectors in GHB_1^C that meet constraint C at equality.

B.2 Proof of Lemma 2

Recall that in Lemma 1, we generated a matrix A of n linearly independent, integer vectors from $span\left(GHB_1^C\right)$. We will show how to add to matrix A one more linearly independent, integer vector from $span\left(GHB_1^C\right)$ for a total of

$(n+1)$ linearly independent integer vectors. Since linearly independent vectors are affinely independent, this will show that $dim(GHB_1^C) \geq n$.

As it stands, matrix A does not have a row with a pivot in component $(2,k)$ (i.e., the k^{th} component of the second block), where $k = (w_b + t - 1)$. But we can generate such a row by creating a vector, $U = (Y - X)$, where Y and X are integer solutions to $SAGHPBC$ and constructed as follows:

In block 1: $Y_{1,k} = X_{1,k} = 1$ all other components are zero

In block 2: $Y_{2,k} = 1 X_{2,k+1} = 1$ all other components are zero

In block $m > 2$: $Y_{m,n} = X_{m,n}$ for all n. Set these binary components in any feasible manner.

Note that, since $U = (Y - X)$, $U_{2,k} = 1$ and all components to the left of $U_{2,k}$ are zero. Because of its unique pivot, this row is linearly independent of the other rows.

B.3 Proof of Lemma 5

Let $w = w_b$.

Case 1: (3.26) with $\tau > t$. Since $Y_{t+w} = 1$, $Y_s = 0$ for all $s > t + w$. Thus, $\tau + w > t + w$ implies that $\sum_{i=\tau+w}^{T} Y_i = 0$.

Case 2: (3.26) with $\tau < t$. Since $X_t = 1$, $X_s = 0$ for all $s < t$ and we have that $\sum_{i=1}^{\tau} X_i = 0$.

Case 3: (3.27) with $\tau < t + w$. Since $Y_{t+w} = 1$, $Y_s = 0$ for all $s < (t + w)$, and we have that $\sum_{i=1}^{\tau} Y_i = 0$.

Case 4: (3.27) with $\tau \geq t + w$. Since $X_t = 1$, we see that $X_s = 0$ for all $s > t$. In particular, $X_s = 0$ for all s.

B.4 Proof of Theorem 7

Let C be an arbitrary constraint of the form (3.10). Then for some time interval t and some flight f, C has the form

$$\sum_{i=t}^{T} Z_i^b - \sum_{i=t}^{T} X_{fi} \leq 0. \tag{B.1}$$

None of our work is affected by the assignment of a flight outside bank b to the last time interval, T. Thus, for ease of vector notation, we can ignore all flights not in bank b and assume that the set of flights, $\{1, 2, \ldots, F + B\}$ is indexed so that variable Z corresponds to flight 1 (i.e., flight 1 is the "ghost flight") and that variable $X_{f\,t}$ corresponds to flight 2.

The proof is almost identical to the proof of Lemma 1. All of the vectors constructed in Algorithm 1 (with $n = n^*$) are in GHB_2^C . All but five of those vectors meet constraint (B.1) at equality. Below are the replacements necessary so that all vectors X and Y generated in (but not output by) algorithm 1 meet (B.1) at equality.

In STEP 1, iteration $j = (t-1)$, change vector Y so that

$Y_{1,t-1} = Y_{2,t-1} = 1$.

In STEP 2, iteration $j = (t-1)$, change vectors Y and X so that

$Y_{1,t+1} = Y_{2,t+1} = X_{1,t+1} = X_{2,t+2} = 1$.

In STEP 3, iteration $j = (t-1)$, change vectors Y and X so that

$Y_{1,t-1} = Y_{2,t-1} = X_{1,t-1} = X_{2,t-1} = Y_{3,t-1} = X3, t = 1$.

With these minor modifications to algorithm 1, all of its output is in the span of the set of vectors that meet (B.1) at equality. Thus, there are at least n^* linearly independent (affinely independent) vectors that meet (B.1) at equality and the face, F, represented by (B.1) must have dimension at least $(n * -1)$.

(B.1) is the one and only constraint to eliminate the solution, X, in which flight f lands in time slot $(t-1)$ and the ghost flight lands in time slot t. Therefore, $dim(F) = (n^* - 1)$, and since $dim(GHB_2^C) = n^*$, F is a facet of GHB_2^C . Moreover, the uniqueness of X implies that (B.1) is the only constraint of its kind that represents F.

B.5 Construction for the proof of Theorem 9

Fix a time interval $t < T$ and let C be the corresponding capacity constraint. Let $k = MAX_{t \neq T}(b_t)$. By re-indexing or adding dummy flights to the set of flights, F , we may assume that the last $(k+1)$ flights of F is a set, F *, of non-bank flights such that $a_f = 1$, for each $f \in F$ *. Let F^* be indexed via $\{f_1, f_2, ..., f_k, f_{k+1}\}$.

Recall that the components of each N^*-dimensional vector are indexed by the set

$$I = \{(i,j) : 1 \leq i \leq F \ \ and \ \ 1 \leq j \leq T\}.$$

We define $I^* \subseteq I$ via $I^* = \{(i,j) \in I : j \neq T\}$. Note that there is one pair in I^* for every component of every flight block except the last component. Thus, $|I^*| = N^* - \ F = n^*$. For each $(i,j) \in I^*$, we will generate a row vector U with a lead "1" (all zeros to the left) in component (i,j). The set Ω will be the collection of all such U-vectors. Thus, $|\Omega| = n^*$ and the vectors in are linearly independent because they can be used to form the rows of an upper triangular matrix.

Fix the index (i,j).

Case 1: $i \in (F - F*)$. First, generate a feasible solution vector, Y as follows. Assign flight i to time interval j. If i is a bank flight in, say, bank b, then let

w_b be the width of the bank b. Assign the flights of $(\Phi_b - \{i\})$ to time intervals $j, j+1, ..., j+w_b$ in any feasible manner that does not exhaust the capacity of interval $(j+1)$ (this uses Assumption 2). If $j = t$, then assign the last $(b_t - 1)$ flights of F^* to time interval t. If $j \neq t$, then assign the last b_t flights of F^* to time interval t. Assign every flight in $(F - \Phi_b)$, including the remaining flights of F^*, to time interval T. Note that whether $j = t$ or $j \neq t$, there are exactly b_t flights assigned to time interval t and that there is exactly one flight assigned to time interval j. Vector Y meet the capacity constraint at equality, hence, is in F_t.

Secondly, generate a feasible solution (row) vector X by setting every component of X as in Y, except that flight i should be assigned to time interval $(t+1)$ (this is possible by assumption 2). If $j = t$, then one of the flights that is currently assigned to time interval t should be reassigned to time interval T. Thus, there will be exactly b_t flights assigned to interval t. Vector X meets the capacity constraint at equality, hence, is in F_t.

Let $U = (Y - X)$. Clearly, U is a linear combination of vectors in F_t. Y and X are the same in all components strictly to the left of (i, j). Moreover, $Y_{i,j} = 1$, $X_{i,j} = 0$, $Y_{i,j+1} = 0$, $X_{i,j+1} = 1$. Thus, $U_{i,j} = 1$, $U_{i,j+1} = 1$ and all other components of U are zero. U has a lead "1" in component (i, j), as desired.

Case 2: $i \in F^* = \{f_1, f_2, ..., f_k, f_{k+1}\}$.

If $i = f_1$, then construct row vectors Y and X as follows.

Vector Y: Assign all flights of $(F - F^*)$ to time interval T. Assign f_1 to time interval j. If $j \neq t$, then assign b_t of the flights of $(F^* - \{f1\})$ to time interval t. This is possible because $|F^*| = (k+1)$, where $k = b_t$. And if $j = t$, assign $b_t - 1$ flights of F^* to time interval t. Either way, the number of flights assigned to time interval t is b_t and the vector Y is in F_t.

Vector X: Assign f_1 to time interval $(j+1)$. If $(j+1) \neq t$, then assign b_t of the flights of $(F^* - \{f_1\})$ to time interval t. And if $j = t$, then assign $b_t - 1$ flights of F^* to time interval t. Either way, the number of flights assigned to time interval t is b_t and the vector X is in F_t.

If $i > f_1$, then construct row vectors Y and X as follows.

Vector Y: Assign all flights of $(F - F^*)$ to time interval T. Assign flight i to time interval j. If $j \neq t$, then assign b_t of the flights of $(F^* - \{i\})$ to time interval t. If $j = t$, then let $d = b_t - 1$ and assign d flights to time interval t. Assign all remaining flights of $(F^* - \{i\})$ to time interval T.

Vector X: For each $i < F_1$ and for each j, let $U = (Y - X)$. Clearly, U is a linear combination of vectors in F_t. Y and X are the same in all components strictly to the left of (i, j). Moreover, $Y_{i,j} = 1$, $X_{i,j} = 0$, $Y_{i,j+1} = 0$, $X_{i,j+1} = 1$. Thus, $U_{i,j} = 1$, $U_{i,j+1} = 1$ and all other components of U are zero. U has a lead "1" in component (i, j), as desired.

B.6 Network Proofs

Lemma 15. *For every $S \geq 2$, the primal matrix (I, A) of the LP relaxation of SGH_2, fails to be a network matrix.*

Proof. Since every submatrix of a network matrix is itself a network matrix, it will suffice to show that A contains a submatrix B that is not a network matrix. We form B by intersecting those rows corresponding to parameters $D1, D2, b_1^1, b_2^1, b_1^2, b_2^2$ and those columns corresponding to variables $Z_1, Z_1^1, Z_1^2, X_1, X_2$. If the rows and columns are arranged in the orders just listed, then B is as below.

$$B = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & -1 & 0 \\ 0 & -1 & 0 & 0 & -1 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & -1 & 0 & -1 \end{bmatrix}$$

By way of contradiction, let us assume that B is a network matrix. Let G be the corresponding undirected graph $G = (V, E)$ and let $e_1, e_2, ..., e_6$ be the basis of edges marked by the identity columns in (I, A) that form a spanning tree of G. Every column of B is the edge-path characteristic vector of a path in G with respect to the edges $e_1, e_2, ..., e_6$. That is, for each column j of B, if we set

$$X_j = \left\{ i : the\ i^{th}\ entry\ of\ column\ j\ is\ nonzero \right\},$$

then the following set P_j of edges is a path in G:

$$P_j = \{e_i : i \in X_j\}.$$

We will derive a contradiction from the co-existence of the paths $P_1, P_2, ..., P_6$ in G. We have that

$$X_4 = \{1, 3, 5\} \quad X_5 = \{2, 4, 6\}$$

and therefore,

$$P_4 = \{e_1, e_3, e_5\} \quad P_5 = \{e_2, e_4, e_6\}.$$

The ordering of the edges in paths P_4 and P_5 is, as of yet, undetermined, but the paths must be isomorphic, respectively, to the ones shown in Figure **??**.

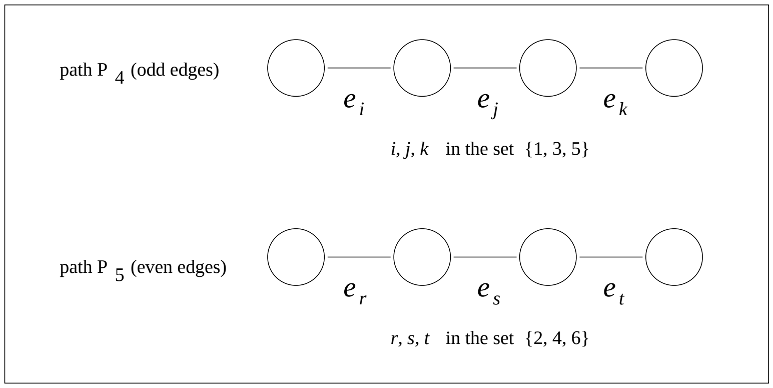

Figure B.1: The two paths

Since G is spanned by a tree with six edges, the total number of nodes in G is $6+1=7$. There are 8 nodes in Figure B.1, so paths P_4 and P_5 must have exactly one node in common. The reader can easily verify that G must be isomorphic to exactly one of the three graphs in Figure B.2.

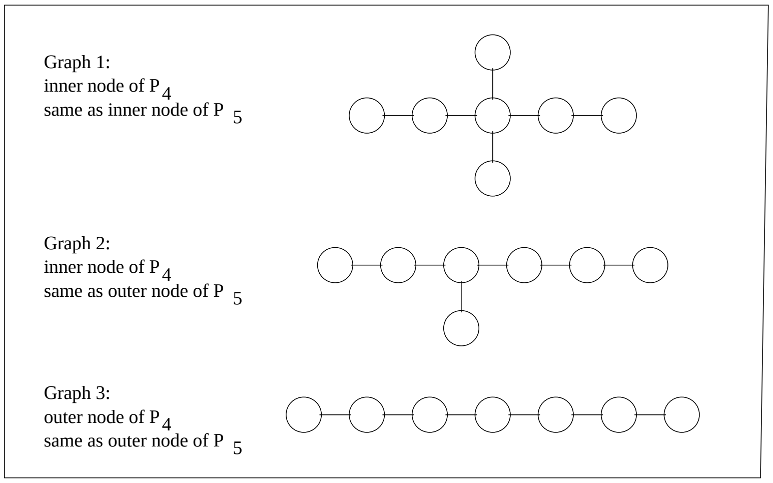
Graph 1:
inner node of P 4
same as inner node of P 5
Graph 2:
inner node of P 4
same as outer node of P 5
Graph 3:
outer node of P 4
same as outer node of P 5

Figure B.2: The three isomorphisms

Fix any one of those graphs G' and consider its left-most edge, e_i. This edge is adjacent to exactly one edge, e_j. The index i is either odd or it is even. We assume that it is odd, the even case being symmetric. Every odd edge lies on path P_4 and is adjacent to another odd edge, so j must be odd. Note that

$$X_1 = \{1,2\} \quad X_2 = \{3,4\} \quad X_3 = \{5,6\}$$

and

$$P_1 = \{e_1, e_2\} \quad P_2 = \{e_3, e_4\} \quad P_3 = \{e_5, e_6\}$$

so every odd edge is adjacent to an even edge and j must be even. This contradicts the previously established parity of j. $\square$

Lemma 16. *The matrix N, as defined in (4.26) is in network matrix form.*

Proof. The matrix N is a $(0,1,-1)$ matrix because it is comprised of $(0,1,-1)$ block matrices. We must show that an arbitrary column c of N contains at most two nonzero entries and that whenever c contains exactly two nonzero entries, they sum to zero. By definition N, there are only two cases to consider.

Case 1: The nonzero entries of c fall within a column of exactly one of the following matrices: Δ, Δ',or ε. Each of these matrices is clearly a NAIM, so the result follows.

Case 2: The nonzero entries of c fall within a column of a matrix defined via

$$M = \begin{bmatrix} (-1)\varepsilon \\ \varepsilon \end{bmatrix}.$$

Each column of ε contains at most one nonzero entry. Thus, M is a NAIM and the result follows. $\square$

Appendix C

Banking Constraint Results

Data Set 1								
79 flights, 1 bank, bank width = 3, cap = 5, T=6								
		LP			IP			
Model	Gap(%)	Value	Iter	Time (s)	Value	Iter	Time (s)	Nodes
XTC	3.41	90.60	87	0.13	93.80	382	0.70	93
XW	1.81	92.10	113	0.13	93.80	209	0.30	18
XWZ	1.81	92.10	113	0.12	93.80	345	0.53	4
XMM	11.64	82.88	111	0.17	93.80	1991	4.75	776
XMMZ	11.64	82.88	111	0.18	93.80	231	0.30	6
XSS	0.00	INT 93.80	92	0.15	93.80	92	0.15	0
WSS	0.00	INT 93.80	172	0.23	93.80	172	0.23	0
XGF	0.00	INT 93.80	93	0.12	93.80	93	0.15	0
WGF	0.00	INT 93.80	158	0.18	93.80	158	0.18	0

Table C.1: Model performances on Data Set 1

Data Set 2								
25 flights, 2 banks (9 each), bank width = 3, cap = 5, T = 6								
		LP			IP			
Model	Gap(%)	Value	Iter	Time (s)	Value	Iter	Time (s)	Nodes
XTC	11.35	100.35	124	0.27	113.20	3773	0.27	629
XW	8.82	103.22	159	0.28	113.20	1352	0.27	156
XWZ	8.82	103.22	159	0.28	113.20	268	0.27	8
XMM	25.22	84.65	118	0.20	113.20	23904	64.10	6161
XMMZ	25.22	84.65	118	0.20	113.20	226	0.42	4
XSS	2.32	110.57	112	0.30	113.20	139	0.42	8
WSS	2.32	110.57	224	0.53	113.20	290	0.82	26
XGF	2.32	110.57	115	0.18	113.20	125	0.27	3
WGF	2.32	110.57	176	0.30	113.20	197	0.35	2

Table C.2: Model performances on Data Set 2

Data Set 3								
79 flights, 2 banks (10 each), bank width = 3, cap = 5, T = 16								
		LP			IP			
Model	Gap(%)	Value	Iter	Time (s)	Value	Iter	Time (s)	Nodes
XTC	0.47	890.92	585	1.30	895.10	2932	20.22	540
XW	2.96	868.62	961	3.42	NL 934.40	388,988	2021.37	20,000
XWZ	2.96	868.62	967	3.93	895.10	1561	6.27	16
XMM	5.37	847.00	596	3.33	NL 920.10	134,303	924.90	20,000
XMMZ	5.37	847.00	596	3.35	895.10	1548	7.38	14
XSS	0.31	892.35	720	8.13	895.10	4645	86.47	574
WSS	0.31	892.35	1937	17.60	895.10	4343	50.88	382
XGF	0.31	892.35	657	3.30	895.10	694	3.40	3
WGF	0.31	892.35	1683	7.68	895.10	1733	10.05	5

NL - node limit reached (20,000)

Table C.3: Model performances on Data Set 3

Data Set 4								
120 flights, 4 banks (8 each), bank width = 3, cap = 5, T = 24								
		LP			IP			
Model	Gap(%)	Value	Iter	Time (s)	Value	Iter	Time (s)	Nodes
XTC	0.69	2784.71	1369	4.73	NL 2915.20	437,068	4171	20000
XW	4.90	2666.72	2525	16.08	NL 2898.20	726,325	8556	20000
XWZ	4.90	2666.72	2525	16.08	2804.10	17502	191	283
XMM	7.22	2601.53	1637	21.18	NL 3001.60	279,618	3801	20000
XMMZ	7.22	2601.53	1637	21.18	2804.10	11980	149	146
XSS	0.26	2796.68	1838	38.95	NL 2804.60	398,102	11,589	20000
WSS	0.26	2796.68	7060	142.38	NL 2804.80	464,154	11,958	20000
XGF	0.26	2796.68	1845	14.22	2804.10	2924	30	24
WGF	0.26	2796.68	4489	26.55	NL 2804.80	5478	64	26
NL - node limit (20,000) reached								

Table C.4: Model performances on Data Set 4

Data Set 5A (13:00-16:59)								
280 flights, 6 banks (12-36 each), bank width = 6, cap = 10, T = 30								
		LP			IP			
Model	Gap(%)	Value	Iter	Time (s)	Value	Iter	Time (s)	Nodes
XTC	0.00	7317.78	1996	29.17	7318.10	2002	30.43	3
XW	0.09	7311.60	6784	103.42	7318.10	8938	156.33	140
XWZ	0.09	7311.60	6784	103.90	7318.10	6805	104.02	4
XMM	0.41	7287.81	2110	55.35	7318.10	61411	3807.30	20000
XMMZ	0.41	7287.81	2110	55.63	7318.10	2225	63.48	3
XSS	0.00	*7318.10	6925	1132.73	7318.10	6925	1127.28	0
WSS	0.00	*7318.10	47,250	9037.80	7318.10	47250	9024.78	0
XGF	0.00	*7318.10	3875	71.82	7318.10	3875	71.63	0
WGF	0.00	*7318.10	12,708	292.30	7318.10	12,708	292.30	0
* Integer solution								

Table C.5: Model performances on Data Set 5A

Data Set 5B (13:00-18:59)								
419 flights, 6 banks (12-36), bank width = 6, cap = 10, T = 42								
		LP			IP			
Model	Gap(%)	Value	Iter	Time (s)	Value	Iter	Time (s)	Nodes
XTC	0.05	14579.28	3533	39.63	14587.10	5276	150.82	323
XW	0.74	14478.50	23,325	381.70	NL 14620.40	326130	7720.70	20,000
XWZ	0.74	14478.50	23,325	381.10	14587.10	23,892	383.90	16
XMM	1.67	14343.19	5224	137.83	NL 14647.60	158201	10559.28	20000
XMMZ	1.67	14343.19	5224	138.15	14587.10	6302	286.23	29
XSS	N/A	N/A	N/A	TLIM	N/A	N/A	TLIM	N/A
WSS	< 0.01	*1458.71	34,701	2137.72	14587.10	34701	2136.73	0
XGF	< 0.01	*1458.71	22,052	1590.52	14587.10	22052	1508.53	0
WGF	< 0.01	*1458.71	34,701	2137.72	14587.10	34,701	2138.42	0

* integer solution TLIM - 3 hour CPU time limit reached time

N/A - not applicable (limits reached) NL - node limit reached (20,000)

Table C.6: Model performances on Data Set 5B

Data Set 5C (13:00-20:59)								
536 flights, 6 banks (12-36), bank width = 6, cap = 10, T = 54								
		LP			IP			
Model	Gap(%)	Value	Iter	Time (s)	Value	Iter	Time (s)	Nodes
XTC	0.06	22822.05	6504	187.93	22835.90	11725	1360.90	1368
XW	0.82	22647.80	49186	1888.05	N/A	N/A	TLIM	N/A
XWZ	0.82	22647.80	49186	1886.98	22835.90	50620	2360.15	35
XMM	1.72	22442.19	7973	531.82	N/A	N/A	TLIM	N/A
XMMZ	1.72	22442.19	7973	533.02	22835.90	11460	892.77	48
XSS	N/A	N/A	N/A	TLIM	N/A	N/A	TLIM	N/A
WSS	N/A	N/A	N/A	TLIM	N/A	N/A	TLIM	N/A
XGF	0.03	22829.87	14944	1197.22	22835.90	15030	1205.31	3
WGF	N/A	22829.87	81633	8266.10	22835.90	81718	8296.67	2

TLIM - 3 hour CPU time limit reached time

N/A - not applicable (limits reached)   NL - node limit reached (20,000)

Table C.7: Model performances on Data Set 5C

Appendix D

Glossary

D.1 General

AAR = arrival acceptance rate (capacity of a time interval)
AOC = airline operational center
ARTCC = air route traffic control center
ATCSCC = Air Traffic Control Systems Command Center, a branch of the FAA
ATFM = air traffic flow management
CDI = capacity demand inequity
CDM = collaborative decision making
CTA = controlled time of arrival
CTD = controlled time of departure
DM = decision maker
ETA = estimated time of arrival
FAA = Federal Aviation Administration
FADE = FAA Airline Data Exchange (program)
FSM = flight schedule monitor
GDP = ground delay program

GH = ground-holding problem

GPS = global position system

IP = integer program

JFK = John F. Kennedy Airport

LP = linear program

LTI = landing time interval

MAGHP = multi-airport ground-holding problem

MAR = managed arrival resevoir

NAIM = node-arc incidence matrix

NAIM = node arc incidence matrix

NAS = national airspace system

NEXTOR = National Center for Excellence in Aviation Operations Research

OAG = official airline guide

OGTA = original gate time of arrival

PAAR = planned arrival acceptance rate

SAGHP = single airport ground-holding problem

SGH = stochastic ground-holding problem

SGH_2 = single airport ground-holding problem, simplfied by preprocesing

TFMP = (air) traffic flow management problem

UPS = United Parcel Service

SGH_2 = simplified version of the stochastic ground-holding problem

D.2 Mathematical

B&B = branch and bound strategy, used in solving integer programs

B-S = Bertsimas Stock variables, (of the form $W_{f\,t}$)

GH = an integer program model of the ground-holding problem

GH_{LP} = the linear program (LP) relaxation of GH

GHB = ground-holding problem with banking constraints

GHB_1 = set of integer solutions to model XSS

GHB_2 = set of integer solutions to model XGF

IP = integer program

LP = linear program

MIP = mixed integer program (integer and non-integer variables)

P^C = convex hull of integer solutions to set

P_{LP} = feasible points in LP relaxation of an integer program P

polytope = a bounded polyhedron

RBS = ration by schedule, an algorithm for alloting arrival slots

TU = totally unimodular (matrix), sq submatrices with det 0,1, or -1

WGF = B-S version of model XGF

WSS = B-S version of model XSS

XGF = ghost flight model of GHB

XMM = monotone marker model of GHB

XMMZ = a relaxation of model XMM (only Z variables declared integer)

XSS = Sdouble sum model of GHB

XTC = time coefficient model of GHB

XW = window marker model of GHB

XWZ = a relaxation of model XW (only Z variables declared integer)

D.3 Constants and Variables

a_f = scheduled time of arrival of flight f

b_t = capacity of time interval t (also called AAR)

b_t^s = capacity of time interval t (AAR) under scenario s

$\widetilde{c}$ = cost of one unit of ground delay

$\widehat{c}$ = cost of one unit of airborne delay

C_f = cost of delaying flight f for one time period

Φ_b = the set of flights in bank b

E_t = number of exempt flights in time period t

F = total number of flights bound for an airport under a GDP

F_t = the face of GHB_1(or GHB_2) defined by capacity constraint t

$n = TF - F$ = the dimension of Euclidean space for model XSS

$n^* = n + (TB - B)$ = the dimension of Euclidean space for model XGF

$N = FT$ = number of components in vectors X feasible to XSS

$N^* = T(F + B)$ = number of components in vectors X feasible to XGF

T = total number of time periods

w_b = desired number of time intervals in which bank b should land

$W_{f\,t}$ = B-S variable. $W_{f\,t} = 1$, if f arrives by time t, $W_{f\,t} = 0$, else.

X = feasible solution to GHB, $X = (X_{11}, ..., X_{1T}, X_{21}, ..., X_{2T}\,, X_{F1}, .., X_{FT})$

X_t = (context: SGH) planned arrival acceptance rate for time period t

$X_{f\,t}$ = assignment variable of flight f to time period t

$\tilde{Z}_t$ = numb of flights held over on the ground from time period t to $t + 1$

$\hat{Z}_t^s$ = numb of flights held in the air from time period t to $t + 1$, in scenario s

BIBLIOGRAPHY

[1] "Aircraft Operating Costs by Stage of Flight and Associated Air Traffic Control Delay Costs Based on a Typical Flight," 1993, *Air Transportation Association*, Washington, 1995.

[2] Ahuja, R., T. Magnanti, and J. Orlin. 1993, *Network Flow: Theory, Algorithms and Applications,* Prentice Hall, Englewood Cliffs, New Jersey.

[3] Andreatta, G., and G. Romanin-Jacur. 1987, "Aircraft Flow Management Under Congestion," *Transportation Science*, **21**, 249-253.

[4] Andreatta, G., A. Odoni, and O. Richetta, O. 1993, "Models for the Ground-Holding Problem," in *Large-Scale Computation and Information Processing in Air Traffic Control*, L. Bianco and A.R. Odoni, eds., Springer-Verlag, Berlin, 125-168.

[5] Attwool, V.W. 1977, "Some Mathematical Aspects of Air Traffic Systems," J. Inst. Navigation., **30**, 394-411.

[6] Ball, M. O., W. Liu, and W. Pulleyblank. 1989, "Two-Terminal Steiner Tree Polyhedra," in *Contributions to Operations Research and Economics: the Twentieth Anniversary of CORE*, MIT Press, Cambridge, Mass, 251-284.

[7] Ball, M., R. Dahl, L. Stone, and T. Thompson. 1993, "OPTIFLOW Build-I Design Document, Version 1.5," *Optiflow Project Report to FAA*, December.

[8] Bertsimas, D. and S. Stock. 1996, "Air Traffic Flow Management Problem with Enroute Capacities," to appear in *Operations Research.*

[9] Birge, J. and F. Louveaux. 1997, *Introduction to Stochastic Programming,* Springer-Verlag, New York, New York.

[10] Bixby, R.E. and W.H.Cunningham. 1980, "Converting Linear Programs to Network Flow Problems," *Math. of Operations Research*, August, 483-508.

[11] Boswek, S.B. and J.E. Evans. 1997, "Analysis of Downstream Impacts of Air Traffic Delay," Project report, MIT Lincoln Lab, March.

[12] Gilbo, E.P. 1993 "Airport Capacity: Representation, Estimation, Optimization," *IEEE Transactions on Control Systems Technology,* **1**, 308-313.

[13] Glockner, G.D. 1996, "Effects of Air Traffic Congestion Delays Under Several Flow Management Policies," Georgia Inst. Tech, dissertation.

[14] Glover, F., D. Klingman, and N. Phillips. 1992, *Network Models in Optimization and Their Applications in Practice*, John Wiley and Sons, Inc., New York, New York.

[15] Hoffman, R. and M. Ball. 1997, "A Comparison of Formulations for the Single-Airport Ground Holding Problem with Banking Constraints," Submitted for publication.

[16] Infanger, G. 1994, *Planning Under Uncertainty: Solving Large-Scale Stochastic Linear Programs.* Boyd and Fraser publising company, Danvers, Massachusetts.

[17] Nemhauser, G.L. and L.A.Wolsey. 1998, *Integer and Combinatorial Optimization*, John Wiley and Sons, Inc., New york, New York.

[18] Odoni, A.R. 1987, "The Flow Management Problem in Air Traffic Control," In *Flow Control of Congested Networks*, A.R. Odoni, L. Bianco, and G. Szego, eds., 269-288, Springer-Verlag, Berlin.

[19] Powell, Warren B., P. Jaillet and A. Odoni. 1995, "Stochastic and Dynamic Networks and Routing," *Handbook in Operations Research and Management Science*, **8**, M. O. Ball, et. al., editors, Elsevier Science, Berlin.

[20] Pulleyblank, W. R.. 1989, "Polyhedral Combinatorics," in *Handbook on Operations Research and Management Science*, *Volume 1: Optimization*, G.L. Nemhauser, A.H.G. Rinnooy Kan, and M.J. Todd, eds., North-Holland, Amsterdam.

[21] Richetta, O. and A.R. Odoni 1993. "Solving Optimally the Static Ground-Holding Policy Problem in Air Traffic Control," *Transportation Science,* **27**, 228-238.

[22] Richetta, O. 1995, "Optimal Algorithms and a Remarkably Efficient Heuristic for the Ground-Holding Problem in Air Traffic Control," *Operations Research*, **43**, 758-770.

[23] Richetta, O. 1991, "Ground Holding Strategies for Air Traffic Control under Uncertainty," PhD dissertation, Massachusetts Institute of Technology.

[24] Sokkapia, B. G. 1985, "Arrival Flow Management as a Feedback Control System," The Mitre Corporation, Washington, D.C.

[25] Terrab, M. and A.R. Odoni. 1993, "Strategic Flow Management for Air Traffic Control," *Operations Research*, **41**, 138-152.

[26] Tutte, W.T. 1960, "An Algorithm for Determining Whether a Given Binary Matroid is Graphic," *Proc. Amer. Math. Soc.*, **11**, 905-917.

[27] Vanchieri, Anthony, R. 1997, "Aviation Dream Team," *ORMS Today*, April, 39-41.

[28] Venkatakrishnan, C., A. Barnett, A. Odoni. 1993, "Landings at Logan Airport: Describing and Increasing Airport Capacity," *Transportation Science,* **27**, 211 - 227.

[29] Vranas, P., D. Bertsimas, and A. R. Odoni. 1994, "The Multi-Airport Ground-Holding Problem in Air Traffic Control," *Operations Research*, **42**, 249-261.

[30] Vranas, P., D. Bertsimas, and A. R. Odoni. 1994, "Dynamic Ground-Holding Policies for a Network of Airports," *Transportation Science,* **28**, 275-291.

[31] Wambsganss, M. 1996, "Collaborative Decision Making Through Dynamic Information Transfer," *Air Traffic Control Quarterly*, **4**, 107-123.

[32] Wang, H. 1991, "A Dynamic Programming Framework for the Global Flow Control Problem in Air Traffic Management," *Transportation Science,* **25,** 308-313.